Universitat Autònoma de Barcelona

escola d'enginyeria

5514: Optimización Multi-Objetivo de la Secuencia de Aterrizaje de Aviones

(Multi-Objective Optimization of Aircraft Landing Scheduling)

Memoria del Trabajo de Fin de Grado
Gestión Aeronáutica
realizado por
Patricia Huguet García
y dirigido por
Liana Napalkova y Juan José Ramos
Sabadell, a 11 de Julio de 2013

Optimización Multi-Objetivo de la Secuencia de Aterrizaje de Aviones

(CERTIFICADO PROYECTO)

Título del Proyecto: Optimización Multi-Objetivo de la Secuencia de aterrizaje de aviones

Autora: Patricia Huguet García **Fecha**: Julio 2013
Tutores: Liana Napalkova y Juan José Ramos
Titulación: Gestión Aeronáutica

Palabras Clave

Castellano: Aterrizaje aviones, Problema de Secuencia de aterrizaje de aviones, Optimización Multi-Objetivo
Catalán: Aterratge avions, Problema de Seqüència d'aterratge avions, Optimització Mutli-Objectiu
Inglés: Aircraft landing, Aircraft Landing Scheduling Problem (ALSP), Multi-Objective Optimization

Resumen del Proyecto

Castellano: Se proponen dos métodos para resolver el problema de Secuencia de Aterrizaje de Aviones. En este problema el controlador debe asignar un tiempo de aterrizaje y una pista a cada avión que entran en el radar, minimizando los costes a la vez que se respetan un número de restricciones operacionales. Los costes asociados con los tiempos de aterrizaje de aviones varían según el tiempo de aterrizaje asignado. Las restricciones hacen referencia a la separación que tiene que haber entre aterrizajes consecutivos, separación que depende del tipo de avión, y a la ocupación de pista. Dado un número de aviones en un tiempo determinado, se proponen dos métodos basados en Algoritmos Evolutivos Multi-Objetivo (MOEAs). Se comparan los dos algoritmos y la calidad de las soluciones obtenidas.

Catalán: Es proposen dos mètodes per resoldre el problema de seqüencia d'aterratges d'avions. En aquest problema, el controlador ha d'assignar un temps d'aterratge y una pista a cada avió que entra dins el rang del radar, minimitzant els costos i alhora respectant un nombre restriccions. Els costos associats amb els temps d'aterratge d'avions varien segons el temps d'aterratge assignat. Les restriccions fan referència a la separació que ha d'haver entre aterratges consecutius, separació que depèn dels tipus d'avió, i a l'ocupació de pista. Donat un nombre d'avions determinat, es proposen dos mètodes basat en Algoritmes Evolutius Multi-Objectiu (MOEAs). Es comparen els dos algoritmes i la qualitat de les solucions obtingudes.

Inglés: Two approaches are described for solving the Aircraft Landing Scheduling

Problem. In this problem, the air traffic controller must assign a landing time for each aircraft which enters the radar range, and must attempt to minimize cost while considering a number of constrains and restrictions. The costs associated with aircraft landing times varying from the preferred landing time. The constrains are concerned with runway occupancy and the separation distance between two consecutive landing aircrafts, which depends on the type of aircraft. Given a certain number of aircrafts, two approaches are proposed based on Multi-Objective Evolutionay Algorithms (MOEAs). Both algorithms are compared and also the quality of the solutions.

Tabla de contenido

Agradecimientos

Quisiera agradecerle a mi tutora Liana Napalkova su dedicación, su labor de dirección y supervisión, gracias a la cual se ha podido llevar a buen término este proyecto.

A mi familia sabiendo que jamás encontraré la forma de agradecerles su constante apoyo y confianza. Sólo espero que comprendan que mis logros y esfuerzos han sido, son y serán suyos e inspirados en ellos.

Finalmente, quiero agradecer a mis amigos, por estos años, por su constante apoyo y la enorme paciencia mostrada durante todo este tiempo.

Lista de Ilustraciones

Lista de Gráficos

Lista de Tablas

1. Introducción

1.1. Interés y motivación

La capacidad de los aeropuertos (y por tanto de pistas de aterrizaje y *stands*) se está convirtiendo cada vez más en un factor que limita la capacidad de cubrir la creciente demanda de vuelos. Esto conlleva retrasos tanto en el tráfico aéreo como en el de tierra. Estos retrasos, para la mayoría de aeropuertos Europeos y de los Estados Unidos, representan de media 15 minutos por avión. Como la construcción de nuevos aeropuertos no es una solución a corto plazo, se han realizado varias investigaciones de cómo hacer un uso más eficiente de los recursos disponibles en el aeropuerto, en general, y de las pistas disponibles, respetando las restricciones de seguridad, en particular.

Se ha demostrado que el problema de encontrar una secuencia de aterrizaje óptima, en el que se tenga en cuenta que la separación entre aterrizajes depende del tipo de avión, es un problema *NP-hard*; es decir, un problema que no tiene solución en un tiempo de computo polinomial.

El objetivo de este proyecto es analizar el problema de secuencia de aterrizaje de aviones y proponer una metodología que permita obtener buenas soluciones, con un tiempo de computación razonable.

1.2. Objetivos

Este proyecto tiene doble finalidad:

(i) Analizar el problema de optimización de secuencia de aterrizaje de aviones (*Aircraft Landing Scheduling Problem* – ALSP)

(ii) Proponer una metodología que permita obtener un conjunto de soluciones lo más adecuado posible.

Estos objetivos se dividen en subobjetivos más específicos:

(i) Descripción del entorno del problema.

(ii) Revisión de trabajos existentes relacionados con el problema.

(iii) Analizar los métodos existentes que tratan de dar solución al problema.

(iv) Plantear un método de resolución del problema.

(v) Analizar los resultados obtenidos.

1.3. Novedad

Para resolver el problema de secuencia de aterrizaje de aviones se han utilizado algoritmos basados en la eficiencia Pareto, el NSGA y NSGA-II. A diferencia de otros métodos utilizados para resolver el problema, el NSGA y NSGA-II resuelven el problema multi-objetivo planteado

en un tiempo computacional razonable, lo que permite obtener soluciones buenas y de forma rápida.

1.4. Valor práctico

Los beneficios del uso práctico de los algoritmos NSGA y NSGA-II para la resolución del problema de secuencia de aterrizaje son:

- Bajo tiempo de ejecución de los algoritmos.
- Consideración de múltiples objetivos.
- Los tiempos de separación de seguridad vienen dados según el tipo de avión.
- No ofrece una única solución, sino que ofrece un conjunto de soluciones válidas.

1.5. Metodología i organización del proyecto

Para analizar el problema, para después buscar una solución, es necesario documentarse sobre el entorno donde se produce el problema. En este caso, en el apartado "Aterrizaje de aviones" se explica el proceso de aterrizaje de aviones y sus implicaciones.

Una vez estudiado el entorno se busca información sobre el problema a tratar y el estado de arte de éste. En el apartado "Planteamiento del problema", se define el problema y se plantea de forma matemática.

Cuando se tiene toda la información teórica necesaria del problema se aborda la parte más técnica en la que se analiza los métodos existentes que tratan de dar solución al problema planteado. Esta clasificación de métodos se realiza en el apartado "Taxonomía de métodos de secuencia de aterrizaje de aviones y software".

El problema que se trata es multi-objetivo, por lo que también es necesario analizar los métodos multi-objetivo existentes para luego desarrollar dos de ellos. Esto se realiza en los apartados "Análisis de métodos multi-objetivo existentes" y "Desarrollo de métodos de secuencia de aterrizaje", respectivamente.

Una vez implementado un modelo que resuelva el problema, se verifique y valide; se analizan los resultados, en el apartado "Experimentos numéricos y resultados".

Finalmente, se explican las conclusiones referentes a la resolución del problema.

2. Aterrizaje de aviones

La capacidad de pista de un aeropuerto es impredecible y está sujeta a cambios durante las operaciones, principalmente por las condiciones temporales y de visibilidad. La capacidad de pista para aterrizaje depende de la categoría del avión y la secuencia de aterrizaje. Esto es debido a que la separación requerida entre aterrizajes consecutivos depende de la categoría de los aviones. La clasificación de aviones generalmente se basa su peso.

Los aviones se aproximan al aeropuerto guiados por los controladores de aproximación en un intervalo de tiempo inferior a 30 minutos antes del aterrizaje. A partir de este momento, el controlador debe crear una flujo de aviones hacia la pista. Debido al tiempo limitado disponible y la alta carga de trabajo de los controladores, difícilmente ningún cambio en la secuencia se puede hacer a esas alturas.

El aterrizaje es la fase final de un vuelo, que se define como el proceso que realiza una aeronave que culmina con el contacto del aparato con la tierra. A continuación se detallan los factores influyentes en el aterrizaje, las etapas del aterrizaje y los agentes involucrados desde que el avión aterriza hasta que vuelve a partir.

2.1. Factores implicados en el aterrizaje

En el aterrizaje hay tres factores influyentes que son: el factor tierra, el factor avión y el factor aire.

El factor tierra consiste en: la comunicación con los controladores de tráfico aéreo (ATC) para reportarse y solicitar autorización de aterrizaje, tener un buen funcionamiento de las radioayudas ILS (*Instrument Lansing System)* y VOR (*VHF Omnidirectional Radio Range)* para dar soporte en el aterrizaje y el estado de la pista e iluminación.

El factor avión, consiste en efectuar todo el procedimiento de maniobras necesarias para lograr una reducción en la velocidad bajando la potencia de los motores y accionando las superficies de control de hipersustentación: *flaps* y *spoilers*, las cuales provocarán que la fuerza de sustentación aumente, y las de profundidad: elevadores o timón de profundidad, para mantener un descenso controlado; tener los trenes de aterrizaje listos y seguir la orientación del ATC.

El factor aire que hace referencia a las condiciones en las que el avión realizará su aproximación y aterrizaje.

2.2. Etapas del aterrizaje

En la maniobra de aterrizaje se distinguen cinco etapas:

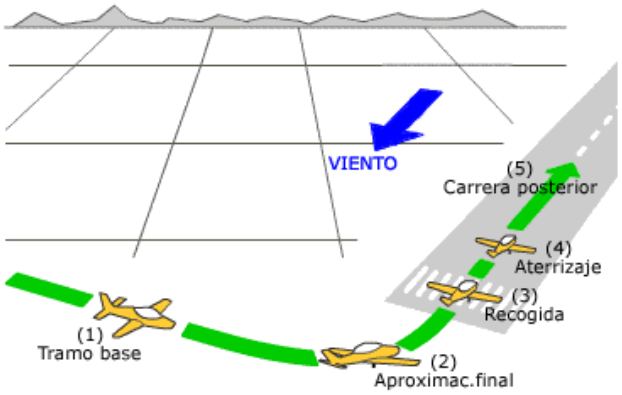


Ilustración 2.1. Fases del aterrizaje

(1) Tramo base. En esta etapa se debe decidir a qué altitud y a qué distancia debe comenzar el descenso y en qué punto se debe virar para estar alineado con el eje de la pista. Estas decisiones influenciarán la calidad del aterrizaje.

(2) Aproximación final. En esta etapa el avión desciende alineada con el eje de la pista. Tiene como objetivos facilitar el contacto con la pista que se realiza en el primer tercio de pista en la cual velocidad no suponga un riesgo de pérdida y proporcione un mínimo de sustentación y velocidad respecto al suelo antes de contactar.

(3) Tramo de recogida. En esta fase el avión hace la transición entre las fases de aproximación y aterrizaje. Comienza cuando el avión se encuentra entre 15 y 20 pies por encima del suelo.

(4) Aterrizaje. Este es el punto donde las ruedas toman contacto con la superficie de aterrizaje y donde todo el peso del avión se transfiere de las alas al tren de aterrizaje.

(5) Carrera posterior. El aterrizaje no concluye hasta que el avión no se frene hasta la velocidad normal de rodadura, o hasta que se detenga totalmente en una zona de parada segura.

2.3. Agentes involucrados en el aterrizaje

En el proceso de aterrizaje del avión y su posterior salida hay tres agentes más, a parte del avión, involucrados: Controlador del tráfico aéreo, las ayudas al atraque, y el handling.

El Controlador de tráfico aéreo es aquella persona encargada de dirigir el tránsito de aeronaves en el espacio aéreo y los aeropuertos, de manera segura, ordenada y rápida. En particular, el controlador debe asignar un tiempo de aterrizaje y una pista, si hay más de una pista en uso. Su labor es complicada, debido al denso tránsito de aviones, a los posibles cambios meteorológicos y a otros imprevistos.

Para mantener la seguridad, en cuanto a separación entre aviones, los ATC aplican normas dispuestas y recomendaciones entregadas por la Organización de Aviación Civil Internacional (OACI), *Federal Aviation Administration* (FAA) y demás autoridades aeronáuticas de cada país.

Normalmente, el grupo de la torre de control lo constituyen personas especializadas en una tarea concreta; por ejemplo, el encargado del radar, el controlador de pistas de aterrizaje y despegue (*Local Control*), el controlador encargado de entregar autorizaciones a las aeronaves que salen bajo reglas de vuelo por instrumentos (*Clearance Delivery*), el controlador encargado de autorizaciones en Calles de Rodaje (*TWY*) y plataforma, (*Ground Control*) o el supervisor general.

Desde que la aeronave abandona la pista hasta su posición de estacionamiento, su trayectoria se acomoda a lo prescrito desde la torre: salida rápida que debe utilizar, detenciones en cruces, accesos a rodaduras paralelas y plataformas designadas y posición final prescrita. A lo largo de este camino existe una señalización horizontal – marcas en el suelo, balizas y luces – y vertical, carteles. Sobre la plataforma de estacionamiento, además, las trayectorias de entrada a la posición, vienen señaladas en el pavimento mediante una línea sobre la que debe rodar el tren delantero. En condiciones de baja visibilidad, señalización incompleta, menor conocimiento del campo de vuelo por la tripulación u otras causas, se utiliza el servicio de coche guía (*followme*), que recibe a la aeronave al final de la calle de salida y la conduce hasta la posición. La aproximación final a la posición de estacionamiento se realiza mediante servicio de señalero u otra alternativa como los sistemas de guía de atraque y los sistemas de ayuda a la guía en atraque. Una vez la aeronave este en posición se detendrá y calzará.

Una vez detenida y calzada la aeronave la asistencia en tierra (*Ground Handling*) prepara a esta para su posterior partida. La asistencia en tierra incluye todos los servicios de que es provista una aeronave desde que aterriza hasta su posterior partida. Los servicios ofrecidos se dividen en:

- Servicio a cabina, incluye todos los servicios dirigidos a dar comodidad a los pasajeros en la cabina del avión así como de la limpieza de la cabina misma.
- Catering, es el abastecimiento de alimentos y bebidas para los pasajeros y tripulación.
- Servicio en rampa, son los servicios en la plataforma de operaciones de la aeronave y también los procesos necesarios para llevar a cabo la carga y descarga de correo, equipaje y demás mercancías a transportar.

- Combustible, servicio de repostaje de combustible.
- Servicios de mantenimiento e ingeniería, incluye todos procesos necesarios para asegurar y mantener la operatividad de las aeronaves. Por su naturaleza es uno de los procesos más delicados en el manejo de una aeronave.
- Servicios de operaciones de campo, es la instancia que coordina a todos los servicios anteriores con el resto de la operación de la aerolínea en el aeropuerto.

2.4. Costes asociados a los retrasos de los aviones

Dentro de las operaciones que se llevan a cabo en el aeropuerto hay ciertas que son críticas, en este caso el aterrizaje y despegue de aviones es una de ellas debido a la limitación de la pista, que es el cuello de botella durante las operaciones del aeropuerto. Por ejemplo, el aeropuerto *London Heathrow*, uno de los más concurridos a nivel mundial, únicamente tiene dos pistas. Cuando el número de aviones aproximándose excede la capacidad del aeropuerto algunos de estos aviones no pueden aterrizar en el tiempo "exacto" previsto. Estos retrasos afectan a compañías aéreas, pasajeros, aeropuerto, slots y compañías de handling.

Aproximadamente el 35% de los retrasos se producen en las fases de pista de salida y entrada. El problema principal de los retrasos es la propagación de éstos que afecta a otros agentes del aeropuerto y a su actividad. Los factores principales causantes de retrasos son: la meteorología, la tripulación, el mantenimiento de los aviones, la congestión aérea, la navegación aérea, los propios pasajeros y las medidas de seguridad.

De estos factores, la navegación aérea causa el 12% de los retrasos. Los retrasos provocados por la navegación aérea hacen referencia a todo aquello implicado en el aterrizaje y despegue de aviones que vienen regidos por las órdenes y autorizaciones de los controladores aéreos.

Con los retrasos se asocian unos costes que afectan a los diferentes agentes que intervienen. Éste coste afecta económicamente a las compañías aéreas, pasajeros, aeropuertos y comunidades y también tiene un coste para las secciones de política, gestión y control de los aeropuertos.

Para las compañías aéreas los costes derivados de los retrasos en las fases de despegue y aterrizaje es principalmente el desperdicio de fuel de cada avión que debe realizar en *holding* o el gasto de fuel al tener que volar más rápido que su velocidad económica.

El tiempo de aterrizaje debe estar comprendido dentro de una predeterminada ventana temporal, limitada por un tiempo mínimo y un tiempo máximo de aterrizaje. Dichas ventanas temporales son diferentes para diferentes aviones. El tiempo mínimo representa el tiempo requerido si el

avión fuese a su máxima velocidad y el tiempo máximo representa la velocidad más económica a nivel de fuel dentro del máximo tiempo permitido.

La velocidad económica de un avión, hace referencia a la velocidad de crucero, ésta varía dependiendo del tipo de avión. El tiempo "objetivo" de aterrizaje de una avión es el tiempo al que debería aterrizar si tuviera que aterrizar a velocidad de crucero. Si el ATC requiere que el avión disminuya, mantenga o acelere la velocidad, se incurrirá un coste. La Figura 2.1. representa la variación del coste de aterrizaje en la ventana temporal del avión.

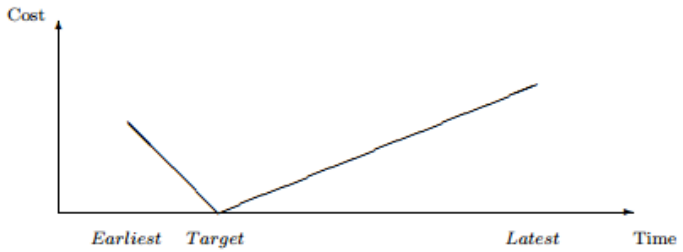


Gráfico 2.1. Variación en el coste de penalización para un avión según el tipo y cantidad de retraso

Dependiendo de la cantidad de retraso, pueden darse casos en los que un número de pasajeros con un vuelo de conexión lo pierdan, que la tripulación del avión tuviera que realizar otro vuelo y éste tiene que ser reprogramado, etc. No únicamente existen los costes de penalización sino que hay muchos otros posibles costes derivados de los retrasos como reprogramaciones de tripulación de tierra, pagos por tiempo extra a la tripulación, reprogramación de vuelos, costes de *handling*, etc. Todos estos retrasos pueden propagarse a otros vuelos, pasajeros, compañías de *handling* y *slots*.

Por lo tanto, resolver el problema de secuencia de aterrizaje (*Aircraft Landing Scheduling Problem - ALSP*), es un problema de asignación a cada avión un tiempo de aterrizaje y pista óptimos de tal forma que se minimice el coste total ya que es un área importante para operaciones de tráfico aéreo.

2.5. Conclusiones

Con el aumento del número de vuelos de las últimas décadas, y por ende enorme aumento de los pasajeros transportados cada día a nivel mundial, debe desarrollarse una coordinación para que los aviones puedan despegar y aterrizar en el número limitado de pistas de las que dispone el aeropuerto respetando las ventanas temporales y las distancias de seguridad de cada avión.

El aeropuerto emite un documento en el que se presentan las operaciones diarias previstas, suministrando las llegadas y salidas de los aviones durante el día. El problema reside en que los vuelos pueden estar sujetos a varios contratiempos y problemas imprevistos (fallos mecánicos, combustible contaminado o muchos otros posibles eventos) que pueden retrasar el vuelo u obligar a un vuelo a tomar una ruta distinta o que deba realizar un aterrizaje de emergencia. Por lo tanto realizar programaciones con mucha antelación no tiene sentido ya que la volatilidad de ésta es elevada.

Como solución se pueden utilizar los datos que proporciona el radar de control de tráfico aéreo. Al entrar en el rango del radar de control de tráfico aéreo del aeropuerto el avión solicita que se le asigne un tiempo de aterrizaje y una pista, en el caso de que el aeropuerto disponga de más de una pista. Por esta razón, es importante desarrollar herramientas que sean capaces de satisfacer las necesidades

El problema de la secuencia de aterrizaje de aviones ALSP es complicado de resolver. Éste puede ser visto como un problema de *Job Machine Scheduling* con en el que los tiempo de procesado y la secuencia son dependientes. El *Job Machine Scheduling* es un problema clasificado como *NP-hard*, de ahí que el problema del secuenciado de aterrizaje de aviones se considere *NP-hard*. El apartado "Taxonomía de Métodos de Secuencia de Aterrizaje de Aviones y Software" se centra en los métodos existentes utilizados para resolver este tipo de problemas.

3. Planteamiento del problema

Al entrar en el rango del radar de control de tráfico de aéreo del aeropuerto el avión solicita que se le asigne un tiempo de aterrizaje y una pista, en el caso de que el aeropuerto disponga de más de una pista.

El tiempo de aterrizaje que se le asigna debe estar comprendido dentro de una ventana temporal específica, limitada por un tiempo *early* de llegada, que representa el tiempo al que puede llegar el avión si va a su máxima velocidad; y por el tiempo *latest* de llegada, que representa el tiempo máximo al que puede llegar un avión si vuela a la velocidad más económica a nivel de fuel.

Existe también la necesidad de imponer que entre dos aviones que aterrizan consecutivos se cumpla una distancia mínima de seguridad. Esta distancia dependerá del tipo de aviones que vayan a aterrizar.

El problema consiste en secuenciar los aviones que entren en el alcance del radar de control de tráfico aéreo de manera que el tiempo de aterrizaje de los aviones, en el mayor número de los casos posibles, esté comprendido dentro de la ventana temporal, y que el tiempo real de aterrizaje sea lo más próximo al tiempo programado de aterrizaje, teniendo en cuenta el cumplimiento de las restricciones de separación de seguridad entre los aviones.

3.1. Notación

A conjunto de aviones

E_i tiempo mínimo de aterrizaje del avión i

T_i tiempo programado de aterrizaje del avión i

L_i tiempo máximo de aterrizaje del avión i

R número de pistas de aterrizaje

S_{ij} separación entre aviones consecutivos si i aterriza antes que j en la misma pista.

s_{ij} separación entre aviones consecutivos si i aterriza antes que j en pistas diferentes.

Z_{max} representa la carga de trabajo en la pista más utilizada.

Z_{min} representa la carga de trabajo en la pista menos utilizada.

x_i tiempo de aterrizaje del avión del avión i

a_i tiempo de retraso del avión i $(x_i > T_i)$,esto es $a_i = \max[0, x_i - T_i]$

b_i tiempo de adelantamiento del avión i $(x_i < T_i)$, esto es $b_i = \max[0, T_i - x_i]$

g_i coste de penalización si el avión i aterriza antes del tiempo programado $(x_i < T_i)$

h_i coste de penalización si el avión i aterriza después del tiempo programado $(x_i > T_i)$

$$y_{ij} = \begin{cases} 1 \; si \; i \; anterriza \; antes \; que \; j \\ 0 \; caso \; contrario \end{cases}$$

$v_{ij} = \begin{cases} 1, & \text{i y j aterrizan en la misma pista} \\ 0, & \text{caso contrario} \end{cases}$

$n_{ir} = \begin{cases} 1, & si\ el\ avión\ i\ \ aterriza\ en\ la\ pista\ r\ (r = 1 \dots R) \\ 0, & caso\ contrario \end{cases}$

$w_{ir} \geq 0$ medida de la carga de trabajo involucrada en aterrizar un avión *i* en una pista *r*.

3.2. Suposiciones

En este planteamiento supondremos que se trata de un caso estático en el que disponemos de toda la información de los vuelos, es decir, disponemos del número de aviones totales involucrados en el problema y éste no varía, y de los datos de cada avión: el tiempo *early*, el tiempo *last*, el tiempo objetivo de aterrizaje y el tipo de avión.

En este caso, consideraremos únicamente una pista.

3.3. Funciones objetivo

El problema se plantea como un problema multi-objetivo en el que se pretende optimizar más de una función objetivo. En este caso las funciones objetivo son:

Minimizar el coste total de la desviación de los aviones. Como se ha mencionado anteriormente, cuando el tiempo de aterrizaje real de un avión no coincide con el tiempo programado incurren costes. En este caso para cada avión se tiene en cuenta el tiempo de retraso y de adelantamiento y los costes respectivos (ya que el coste de retraso no es el mismo que en el caso que el avión llegue antes).

$$Min = \sum_{i=1}^{|A|} (a_i h_i + b_i g_i)$$

Minimizar el tiempo total de llegada de los aviones. Al mismo tiempo se pretende que el tiempo total del proceso de aterrizaje de los aviones sea el mínimo posible, esto es, que el ultimo avión aterrice lo antes posible.

$$Min = \sum_{i=1}^{|A|} x_i$$

Minimizar la diferencia de carga de trabajo en diferentes pistas. En este caso se trata de equilibrar la carga de trabajo de cada pista, para que no se produzca saturación por un lado y ociosidad por el otro, esta función es muy útil a la hora de equilibrar cargas de trabajo, pero en este caso no la aplicamos.

$$Min\ \ Z_{max} - Z_{min}$$

3.4. Restricciones

Ventanas temporales. Asegurar que el tiempo de aterrizaje del avión está comprendido en los tiempos que conforman la ventana temporal.

$$E_i \leq x_i \leq L_i$$

Relaciones de precedencia.

$$y_{ij} + y_{ji} = 1, \quad i, j \epsilon A \quad j > i$$

Separación entre aviones. Esta restricción asegura el cumplimiento de la separación de seguridad exigida entre el aterrizaje consecutivo de dos aviones. En la Tabla 4.1. se representa los tiempos de separación (en segundos) entre aviones que aterrizan consecutivamente.

	Avión siguiente		
1º avión a aterrizar	Heavy	Large	Small
Heavy	96	157	196
Large	60	69	131
Small	60	69	82

Tabla 3.1. Tiempos mínimos de separación (en segundos) entre aterrizajes

$$x_j y_{ij} \geq x_i y_{ij} + S_{ij} v_{ij} + s_{ij}(1 - y_{ij})$$

Restricciones de pista. En el caso que el aeropuerto disponga de más de una pista, asegura que el avión aterriza únicamente en una pista. Esta restricción contempla que si dos aviones consecutivos aterrizan en la misma pista, deben respetar las distancias de separación de seguridad. En el caso contrario, que dos aviones consecutivos aterricen en pistas diferentes, esta distancia de seguridad será 0.

$$\sum_{r=1}^{R} n_{ir} = 1 \quad r = 1 \dots R \,, i = 1 \dots |A|$$

Equilibrio de cargas de trabajo en las pistas. Asegura que la carga máxima de trabajo es como mínimo la carga de trabajo en la pista más utilizada y que la carga mínima es la carga de trabajo pista menos utilizada.

$$Z_{max} \geq \sum_{i=1}^{|A|} w_{ir} n_{ir} \,, r = 1 \dots R$$

$$Z_{min} \leq \sum_{i=1}^{|A|} w_{ir} n_{ir}$$

3.5. Conclusiones

El *Aircraft Landing Scheduling Problem* se plantea como un problema multi-objetivo. En este caso, las consideraciones que se tienen en cuenta son que únicamente se dispone de una pista y que el problema es estático. El planteamiento del problema tiene por funciones objetivos minimizar la desviación total entre el tiempo de aterrizaje y el tiempo objetivo de cada avión y minimizar el tiempo de aterrizaje del último avión, cumpliendo con las restricciones de ventanas temporales de cada avión, las distancias de separación de seguridad entre aterrizajes consecutivos y la restricción de pista.

4. Taxonomía de Métodos de Secuencia de Aterrizaje de Aviones y Software

El objetivo de esta sección es clasificar los métodos y software existentes destinados al aterrizaje de aviones y su programación. La clasificación se lleva a cabo mediante la taxonomía de dichos métodos, para analizar el avance y demostrar los inconvenientes de los enfoques existentes.

La taxonomía de los métodos se lleva a cabo tres fases. Primero se realiza una lista de la clasificación de los parámetros; segundo, se definen los posibles valores de cada parámetro y finalmente, se analizan los métodos descritos en publicaciones científicas sujetos a la clasificación de los parámetros anterior.

4.1. Especificación de los parámetros de clasificación

1. Tipo de problema de optimización

1.1. Objetivo único (*Single-Objective*). La optimización se basa en único objetivo que también puede ser formulado como una suma ponderada de múltiples objetivos.

1.2. Multi-objetivo. Optimización simultánea de más de un objetivo con la finalidad de encontrar un conjunto de soluciones (*equilibrada*)

2. Tipo de funciones objetivo

2.1. Minimizar el *makespan* (tiempo de aterrizaje del último avión) de la secuencia de aterrizaje.

2.2. Minimizar el coste total de la desviación entre el tiempo de aterrizaje final de todos los aviones y su tiempo de aterrizaje estimado.

2.3. Minimizar el tiempo total de la desviación de los aterrizajes de los aviones.

2.4. Minimizar el retraso total de los aviones.

3. Restricciones

3.1. *Uso de pista.* Cada pista puede ser utilizada como máximo por un avión.

3.2. *Número de pistas de las que dispone el aeropuerto.*

3.3. *Separación entre aviones.* Hace referencia a las distancias y tiempos de seguridad que hay que dejar entre aviones a la hora de aterrizar, por razones aerodinámicas y de seguridad.

3.4. *Ventanas temporales de cada avión.* A cada avión se le ha asignado un tiempo estimado de aterrizaje, pero existe un tiempo anterior *Early Time of Arrival* (suponiendo que el avión vaya a máxima velocidad) y un tiempo posterior de aterrizaje

Latest Time of Arrival (determinado por la cantidad de combustible disponible o por el máximo retraso permitido para un avión).

3.5. *Precedencia entre aviones.* Se considera la precedencia de aviones en las secuencias de aterrizaje. Una de las fuentes de esta restricción consiste en las diferentes rutas en las cuales el avión puede acceder al aeropuerto.

3.6. *Time Shifting*. Existe una flexibilidad limitada en cuanto al tiempo de aterrizaje del avión, dentro de su ventana temporal, tanto hacia adelante como hacia atrás en relación al tiempo de aterrizaje estimado.

3.7. Asignación de pista. Esta restricción es específica para problemas que se plantean con más de una pista de aterrizaje. Con ella se asegura que el avión aterriza en una única pista.

4. Heurísticas utilizadas

4.1. *Control.* Son heurísticas que se utilizan para asegurarse que los métodos que se implementan cumplen con las restricciones impuestas al problema.

4.1.1. *Force feasible*. Fuerza que el tiempo de aterrizaje entre aviones sea al menos el mínimo tiempo permitido. Para ello, primero los aviones son ordenados, luego los primeros dos aviones se examinan y si existe un conflicto el resto de la secuencia se atrasa hasta que ya no existe conflicto. Se sigue con la siguiente pareja de aviones, y este proceso continúa secuencialmente hasta que el tiempo de aterrizaje sea al menos el mínimo permitido.

4.1.2. *Squash.* Esta heurística optimiza localmente la secuencia mientras mantiene el orden de caída. Si el tiempo entre aterrizajes de dos aviones es mayor que el tiempo mínimo requerido, seria innecesario atrasar el siguiente, por eso mover el siguiente avión hacia adelante disminuirá el coste.

4.2. *Proposal Generation Mechanism.* Una vez se obtiene una secuencia de aterrizaje, ésta es enviada a un agente (avión) el cual evaluará dicha secuencia y reenviará un conjunto de aviones con los cuales se puede cambiar de pista. Si al intercambiar las pistas de aterrizaje de dos aviones el tiempo total de la secuencia de caída disminuye, entonces se procede a hacer el cambio.

4.3. *Negotiation Strategy.* Las estrategias de negociación permiten seleccionar secuencias de aterrizaje dentro de un intervalo de propuestas proporcionadas por los agentes.

5. Consideraciones

5.1. Entorno estático, en el cual se ordenan un número fijo de aviones de los cuales se conoce toda la información de antemano.

5.2. Entorno dinámico, en el cual se incorporan nuevos aviones que entran en el sistema sin tener en cuenta aquellos que ya han aterrizado; es decir aquellos que ya han salido del sistema.

6. Secuencias de permiso de aterrizaje

6.1. *First-come, First-served* (FCFS). Este algoritmo determina la secuencia de aterrizaje del avión basándose en el orden del *estimated time of arrival* de cada avión.

6.2. *Constrained Position Shifting heuristic*. Esta heurística evita que las posiciones finales de la secuencia de aterrizaje difiera del orden establecido por el FCFS no más de un número pre-especificado, llamado *Maximum Position Shift* (MPS). Es más, cuando el MPS es pequeño, se mantiene imparcial entre los aviones para no desviarse mucho de la secuencia FCFS. *Relative Position Shifting* (RPS) es una variante del MPS que tiene en cuenta la cercanía del avión a la pista cuando se especifica el MPS. Esta heurística assume que el tiempo mínimo de aterrizaje debe ser menor que el tiempo estimado de aterrizaje.

6.3. *Non-dominated Average Ranking*. Este método evalúa la bondad de una potencial secuencia de caída para cada uno de los objetivos y utiliza una estrategia de clasificación para elegir la secuencia más prometedora.

6.4. Heurística de construcción. Heurística para la obtención de una solución inicial basada en la observación siguiente: Si se asume que el número de pistas r >1 y que todos los aviones $T_i(i = 1, \dots, n)$ son ordenados de manera ascendente, es decir $T_{i_1} \leq T_{i_2} \leq \cdots T_{i_{n-1}} \leq T_{i_n}$. Para cada dos aviones ordenados i_ky i_{k+1} si $T_{i_{k+1}} < T_{i_k} + S_{i_k i_{k+1}}$, entonces una secuencia de aterrizaje mejor, que no necesariamente optima, puede encontrarse si se les asigna pistas de aterrizaje diferentes.

7. Modelos de Optimización

7.1. *Job Shop Scheduling Model*. Este modelo se basa , de manera general, en ordenar un conjunto de actividades a ser procesadas en un conjunto de máquinas tal como:

- Cada actividad está compuesta por una secuencia predefinida de operaciones.
- Cada trabajo debe ser precedido por algunas máquinas en un orden específico y no necesariamente en el mismo orden para todas las actividades.
- Cada máquina puede procesar únicamente una actividad cada vez.

Se considera que las pistas son las máquinas y que el aterrizaje de un avión es una operación de un trabajo (avión). Se define un orden entre operaciones para cada trabajo; suponiendo eso, cuando las ventanas temporales de dos aviones están separadas, existe un orden entre ellos, de ahí la apariencia de un orden parcial de los

trabajos a nivel operacional. El objetivo es determinar la secuencia de aterrizaje de los aviones en unas determinadas pistas. Esta tarea consiste en asignar un tiempo de inicio para cada aterrizaje y adjudicar una pista respetando las restricciones de tiempo:

- El tiempo mínimo y el tiempo máximo del aterrizaje.
- Tiempo de separación entre aviones en una misma pista.
- La secuencia de caída.

7.2. *Mixed Integer Programming Model* (MIP). La programación lineal es un procedimiento o un algoritmo matemático mediante el cual se resuelve un problema indeterminado, formulado a través de un sistema de inecuaciones lineales, optimizando la función objetivo, también lineal. Un problema lineal mixto consiste en un modelo LP con algunas variables enteras.

7.3. *Dynamic Programming Model*. Este modelo se basa en transformar un problema complejo en una secuencia de problemas más simples. Primero se resuelven los más sencillos que después servirán para resolver los más difíciles, hasta que todos estén resueltos. Este modelo no está restringido por ningún requerimiento de linealidad, convexidad ni continuidad.

7.4. *Network Model.* En este modelo cada fase corresponde a una posición de un avión en la secuencia final de aterrizaje. Un nodo de una cierta fase de la red representa una sub-secuencia de aviones. El nodo de inicio y el nodo final representan el inicio y el final del proceso de secuencia respectivamente. La finalidad es encontrar el camino más corto en la red, lo que se traduce en, encontrar la secuencia con el menor makespan.

4.2. Recensión de los Trabajos Revisados Sujetos a la Clasificación de Parámetros

Artículos de investigación de las últimas tres décadas han sido analizados sujetos a los parámetros de clasificación especificados. La Tabla 3.1. resume los métodos revisados que más adelante son descritos con mayor detalle.

Parámetros / Métodos	1. Tipo de problema de optimización	2. Tipo de funciones objetivo	3. Restricciones	4. Heurísticas utilizadas	5. Consideraciones	6. Secuencias de permiso de aterrizaje	7. Modelo de optimización
Linear Programming Based Tree	1.1	2.2	3.3 3.4 3.5		5.1		7.2 7.4

Search							
Ant Colony Optimization Genetic Algorithm	1.1	2.2	3.2 3.3 3.4 3.5		5.2		7.1
Genetic Algorithm	1.1	2.3	3.3 3.4 3.5	4.1.1 4.1.2	5.1	6.2	7.2
Branch and Bound Algorithm	1.1	2.3	3.2 3.3 3.4		5.1		7.2
An any time Algorithm	1.1	2.4	3.2 3.3 3.7		5.2		
Multi objective Neighborhood Search Differential Evolution	1.2	2.1 2.4	3.3 3.4 3.5		5.1	6.2	
Semantic Agents Negotiation Mechanism	1.2	2.1 2.4		4.2 4.3		6.3	
Dynamic programming on a Network Model	1.1	2.1	3.3 3.4 3.5 3.6			6.2	7.3 7.4
Hybrid meta-heuristic	1.1	2.2	3.3 3.4 3.7		5.1	6.4	7.2

Tabla 4.1. Clasificación de métodos de resolución del ALSP

- ***Linear Programming Based Tree Search*** [*1*]

  Para resolver el problema, se ha planteado como un *Displacement Problem*. Este problema surge cuando se tiene que realizar una secuencia de decisiones y cada decisión sucesiva que se toma está relacionada con la decisión anterior. Se resuelve el problema estático original y se considera que esta solución es una solución factible (encontrada por algún algoritmo). Una vez obtenida esta solución factible, se considera que *algo* en el problema original cambia, un dato (relacionado con las funciones objetivo), o una nueva variable, o una nueva restricción. Ahora, entorno operacional es diferente a lo que se había asumido previamente al llegar a esas decisiones. Como resultado de este cambio de entorno, es necesario resolver el problema inicial, incorporando algunos cambios, pero con alguna restricción adicional (*link*) que haga referencia a la solución original (previa). Para esto, se define una *displacement function* que cuantifica el efecto de desplazar cada variable de decisión del valor de solución previa (conocido) a su nuevo valor (actualmente desconocido).

  Se considera que el problema es dinámico (5.2) por lo cual el entorno operacional del problema va variando. El método busca optimizar una única función objetivo (1.1), se

busca minimizar el coste total de las desviaciones de los aviones (2.2). La función objetivo del problema, tiene en cuenta los costes de penalización tanto de los aterrizajes antes de hora, y los aterrizajes después del tiempo estimado de aterrizaje. Las restricciones que se contemplan son: la separación de seguridad entre aterrizajes de aviones (3.3), las ventanas temporales de cada avión (3.4) y la precedencia entre aviones (3.5).

Para resolver el problema ALSP como un *Displacement Problem* se propone un algoritmo solución basado en programación linear mixta (7.2) junto a una estrategia de búsqueda de árbol en un grafo (7.4).

Descripción paso- a- paso del algoritmo:

Notación:

- P número de aviones a ser secuenciados;
- i es el índice de un avión;
- γ es el contador de iteraciones;
- X_i es el tiempo de aterrizaje del avión i;
- A_i is an appearance time of an aircraft at which it was first available to be assigned a landing time;
- Z_{disp} es el coste acumulado del desplazamiento;
- Z_{sol} es el coste de la solución final en términos de función de coste asociada al ALS original (estático);
- t^* is a freeze time; any aircraft assigned a landing time within t^* of the current time had its landing time (and runway) frozen;
- $F_0(t)$ represents the set of aircraft that have not yet appeared by time t;
- $F_1(t)$ represents the set of aircraft that have appeared by time t, but have not yet landed (or had their landing times frozen);
- $F_2(t)$ represents the set of aircraft that have appeared by time t and have landed (or have had their landing times frozen);
- g_i es el coste de penalización (≥ 0) por unidad de tiempo por aterrizar antes del tiempo estimado T_i para el avión i.
- h_i es el coste de penalización (≥ 0) por unidad de tiempo por aterrizar después del tiempo estimado T_i para el avión i. ... ;
- λ_{cost} es la importancia asociada al coste total de la solución Z_{sol}
- λ_{disp} es la importancia asociada al coste total del desplazamiento;
- λ_{max} es la importancia asociada al desplazamiento máximo D_{max};
- D_{max} es el desplazamiento máximo...

Algoritmo 1.

(1) Se fijan los tiempos de aterrizaje $X_i=\infty$, $i=1,\ldots,P$.

(2) Sea:
$F_0(t) = [i|A_i \geq t, i = 1, \ldots, P\ ]$
$F_1(t) = [i|A_i \leq t \text{ and } X_i > (t + t^*), i = 1, \ldots, P\ ]$
$F_2(t) = [i|A_i \leq t \text{ and } X_i \leq (t + t^*), i = 1, \ldots, P\ ]$

(3) Se asigna $\gamma = 0$ y $Z_{disp} = 0$. Se fija el tiempo actual $t_0 = \min[A_i \mid i = 1, \ldots, P\ ]$. Y se resuelve el problema ALSP estático original, incluyendo únicamente a los aviones en $F_1(t_0)$. La solución a este problema estático son los tiempos iniciales de aterrizaje $X_i \forall i \in F_1(t_0)$.

(4) Si hay aviones por aparecer $|F_0(t_\gamma)| \geq 1$ entonces ir al paso (5), sino $|F_0(t_\gamma)| = 0$ todos los aviones han aparecido en tal caso ir al paso (6) .

(5) Asignar $\gamma = \gamma + 1$. Avanzar el tiempo a $t_\gamma = \min[A_i \mid i \in F_0(t_\gamma - 1)]$ y resolver el *displacement problem* involucrando únicamente a aquellos aviones en $F_1(t_\gamma) \cup F_2(t_\gamma)$, donde los aviones en $F_2(t_\gamma)$, están restringidos a aterrizar al tiempo $X_i \forall i \in$

$F_1(t_\gamma)$ y en la pista adecuada, como han sido establecido por la solución anterior. Añadir el componente de coste de desplazamiento $(\lambda_{disp} \sum_{i \in F_1(t_\gamma) \cap F_1(t_\gamma - 1)} p_i D_i(\underline{X}, \underline{x}) + \lambda_{max} D_{max})$ de esta solución a Z_{disp} e ir al paso (4).

(6) Todos los aviones en $F_1(t_\gamma)$ se estima que aterricen a los tiempos de aterrizaje (y en la pista correspondiente) como ha sido establecido por la última solución del desplazamiento. Computar $Z_{sol} = \sum_{i=1}^{P} (g_i \, max[0, T_i - X_i] + h_i \, max[0, T_i - X_i])$.

- ***ACOGA (Ant Colony Optimization Genetic Algorithm)* [*ALSP basado en Job Shop scheduling problem* [2]**

  Para resolver el ALSP se propone un método híbrido, llamado ACOGA (*Ant Colony Optimization Genetic Algorithm*). Se considera que el problema es estático (5.1), de manera que el número de aviones con el que se trabaja es fijo desde el inicio. Para resolver el problema lo separa en dos actividades. La primera es obtener una población inicial y la segunda computar los tiempos de aterrizaje para un conjunto de aviones. Este método está basado en un *Job Shop Scheduling model* (7.1) en el que se dividen los aterrizajes por grupos (de como mínimo 2 aviones consecutivos).

  ACOGA es un método de optimización con una única función objetivo (1.1). En este caso, la función objetivo a minimizar es el coste total de la desviación entre el tiempo de aterrizaje actual y el tiempo de aterrizaje estimado (2.2). Dicha función tiene en cuenta tanto el coste por adelantamiento como por retraso del tiempo de aterrizaje del avión respecto al tiempo estimado de aterrizaje.

  Las restricciones que contempla este método son el número de pistas las cuales dispone el aeropuerto (3.2), la separación entre aviones (3.3), las ventanas temporales de cada avión (3.4) y la precedencia entre aviones (3.5).

  En este caso, no se aplica una estrategia de aterrizaje a la hora de definir una secuencia de aterrizaje, sino que se aplica el algoritmo *Ant Colony Optimization* (ACO). La población inicial juega un papel importante a la hora de determinar la calidad de la solución final. La estrategia que utilizan para obtener la población inicial es generar una secuencia de aterrizaje y una asignación de pistas aplicando un ciclo del algoritmo ACO (descrito más adelante). Una vez construida la primera población, se computa el tiempo de aterrizaje de cada avión respetando las ventanas temporales y la separación entre aviones. El siguiente paso es aplicar un algoritmo genético para alterar la población inicial y llegar a una solución óptima.

Descripción paso- a- paso del algoritmo:

Notación:

- m : número de hormigas
- $Tabu_k$ es la lista de nodos visitados;
- $Candidate_k$: lista de nodos candidatos a ser visitados en orden, para respetar la relación de precedencia.
- α , β : parámetros que controlan la importancia relativa de η y τ, donde η es la información de la heurística y τ es el rastro de feromona
- q : variable aleatoria uniformemente distribuida entre [0,1]
- q_0 : parámetro fijado entre 0 y 1
- τ_{ij} es el valor del rastro de las feromonas
- τ_0 : valor inicial de los rastros de feromonas
- ρ : coeficiente de vaporización
- D : lista de nodos del grafo

Algoritmo 2. ACO para la obtención de la población inicial

(1) **Sea:**
- $Tabu_k = D$,
- $Candidate_k$ = *primera operación de J*
- $\tau = \tau_0$

(2) Para cada hormiga k= 1… m, inicializar $i \leftarrow D$.

(3) Mientras $Candidate_k \neq 0$,

(3.1) de un nodo *i*, elegir el siguiente nodo $j \notin Tabu_k$ de entre los nodos de $Candidate_k$ de auerdo con:

$$j = \begin{cases} arg\,max\left[(\tau_{il})^{\alpha}(\eta_{il})^{\beta}\right] & si\ q \leq q_0 \\ l \notin Tabu_k & \\ J\ , si\ q \leq q_0 & (escogida\ aleatoriamente\ entre\ Candidate_k \end{cases}$$

$$\eta_{il} = \begin{cases} \dfrac{1}{(|ta_l - ta_i| * S_{il}) + 1}\ if\ i\ and\ l\ on\ the\ same\ runway \\ \dfrac{1}{(|ta_l - ta_i| * s_{il}) + 1}\ if\ i\ and\ l\ on\ diferent\ runway \end{cases}$$

(3.2) Insertar j en $Tabu_k$. Actualizar la lista $Candidate_k$ y actualizar localmente los rastros de feromonas τ_{ij} utilizando :

$$\tau_{ij} = (1 - \rho l)\tau_{ij} + \rho l \tau_0$$

Note: En el algoritmo previo, la información heurística (η_{il}) depende de dos parámetros:

- Tiempo de separación (S_{il}) or (s_{il}) : generalmente, $s_{il} \ll S_{il}$ $(\forall i, l = 1, \dots, N)$. De esta forma, los aviones que aterricen diferentes pistas serán más privilegiados que aquellos que aterricen en una misma pista.
- $|ta_i - ta_j|$: los aviones cercanos en términos de tiempo objetivo son privilegiados para aterrizar los más cercano posible a su tiempo estimado de aterrizaje para asi reducir el coste de penalización.

- ***Genetic Algorithm*** **[***3***]**

El método se basa en un Problema Programación Lineal Mixto (7.2), con variables reales y enteras. Se considera que el problema es estático (5.1), de manera que el número de aviones con el que se trabaja es fijo desde el inicio.

El problema está planteado con una única función objetivo (1.1), que consiste en minimizar la suma de la ponderación de las desviaciones de los aviones (2.3). Las restricciones que se

contemplan son: la separación entre aviones (3.3), las ventanas temporales de cada avión (3.4) y la precedencia entre aviones (3.5).

Este método utiliza un algoritmo genético que luego servirá para establecer el límite superior para la solución. En este problema, los genes son los aviones y el alelo es el tiempo de aterrizaje del avión. Por lo tanto, el cromosoma representará una tabla de aviones y su tiempo de aterrizaje propuesto. Esto constituye una secuencia de aterrizaje *i*, con una función *fitness* F_i. El algoritmo genético elige a dos padres según sean más aptos, y con ellos realiza un intercambio genético con el que obtiene dos hijos, en este caso un hijo es una secuencia de aterrizaje. Se puede dar el caso que las secuencias resultantes no cumplan las restricciones de las ventanas temporales de los aviones, ni la separación de seguridad que tiene que haber entre dos aterrizajes consecutivos. Estos dos factores hacen que surja la necesidad de heurísticas para procesar el resultado después del intercambio genético. Estas heurísticas son la *Force feasible* (4.1.1) y *Squash* (4.1.2).

Descripción paso- a- paso del algoritmo:

Notación:

- p ,es la población;
- F_i , función de idoneidad;
- S_i , secuencia de aterrizaje resultante después del crossover entre dos progenitores;
- g_i , es el gen que representa un avión;
- $a_{i,}$, es el alelo que representa el tiempo de aterrizaje del avión;
- P_{mut} , es la probabilidad de ser mutado;
- M, es la mutación que se aplica a una secuencia en la que se introducen nuevas características a una población, es este caso, se modificara el orden de aterrizaje de los aviones;

Algoritmo 3 Algoritmo Genético

(1) Se genera una p inicial de forma aleatoria.
(2) Seleccionar dos padres según su F_i. Aquellos que tengan mayor F_i tendrán más posibilidad de ser seleccionados.
(3) Se realiza la combinación y se aplican las heurísticas de *Force feasible* y *Squash*. Como resultado se obtiene S_{i_1} y S_{i_2}
(4) Según P_{mut} se le aplica M a S_{i_1} y S_{i_2}, aleatoriamente modificando el a_i de *gi.* Si esta modificación ha beneficiado a la población aumenta la F_i asociada, en caso contrario disminuye.

- ***Branch & Bound Algorithm*** *[3]*

El método se basa en un modelo de *Mixed Integer Programming* (7.2). Se considera que el problema es estático (5.1), de manera que el número de aviones con el que se trabaja es fijo desde el inicio.

El problema está planteado con una única función objetivo (1.1), que consiste en minimizar la suma de la ponderación de las desviaciones de los aviones (2.3). Las restricciones que se

contemplan son: la separación entre aviones (3.3), las ventanas temporales de cada avión (3.4) y la precedencia entre aviones (3.5).

El problema se resuelve mediante la combinación algoritmo *Branch & Bound* y del algoritmo genético, los cuales proporcionan una solución exacta del problema. El algoritmo *Branch & Bound* utiliza la solución proporcionada por el algoritmo genético y lo utiliza para establecer los límites superiores del problema. Para establecer los límites inferiores del problema, se utiliza el algoritmo simplex. Para buscar conflictos entre dos aviones en un nodo se utiliza una estrategia de búsqueda en profundidad.

Descripción paso- a- paso del algoritmo:

Notación:

- X , es la solución, es decir, la secuencia de aterrizaje;
- t_i^p , es el tiempo preferred del avión i;
- X_i , representa un avión, i= 1,…,N;
- δ_{jk} , representa la restricción de precedencia entre aviones;

Algoritmo 4. Branch & Bound

Dada una solución X para un nodo K
(1) Se ordenan los aviones X_i según su t_i^p
(2) Se comprueba que la restricción de tiempos de separación entre aviones se cumpla
  (2.1) Si la restricción de cumple, se considera que X es una solución factible para el nodo K
  (2.2) Sino, se coge la pareja *(j,k)* y se ramifica a partir de aquí asignando δ_{jk} 1 o 0 (con esto se relaja la restricción de precedencia)

- ***Genetic Algorithm (Standard Binary Algorithm and Seeding Approach)*** [*4*]

Este método resuelve el ALSP mediante dos algoritmos genéticos, los que luego serán comprados.. Se considera que el problema es dinámico (5.2), de manera que el número de aviones con el que se trabaja no está determinado, por lo tanto los constantes cambios debido a despegues y aterrizaje de aviones afectan a las restricciones.

En este caso se optimiza una única función objetivo (1.1) que consiste en minimizar el retraso total de los aviones (2.4). Dicha función tiene en cuenta tanto el coste por adelantamiento como por retraso del tiempo de aterrizaje del avión respecto al tiempo estimado de aterrizaje y aplica una penalización. Las restricciones que contempla este método son el número de pistas las cuales dispone el aeropuerto (3.2), la separación entre aviones (3.3) y las ventanas temporales de cada avión (3.4) .

Para resolver la secuencia de aterrizaje, se plantea el algoritmo genético estándar (*Standard Binary Genetic Algorithm*) y una variación de este. En el primero, cada problema en la secuencia requiere un nuevo cromosoma codificado que elimina los aviones que han aterrizado e incorpora los nuevos aviones que han llegado. La nueva programación se construye desde cero, partiendo desde una población inicial aleatoria. El segundo algoritmo

es una modificación, en la que se eliminan los aviones que han aterrizado y se actualizan los nuevos aviones, pero que, a diferencia de la primera, no construye una nueva programación partiendo de una población inicial aleatoria.

Descripción paso- a- paso del algoritmo:

Notación:

- p , es la población que representa la secuencia de aterrizaje de los aviones,
- g_i son los aviones;
- t son los intervalos de tiempo, cada intervalo es de 3 minutos;

Algoritmo 5. ***Standard Algorithm***

(1) Se inicializa t.
(2) Se inicializa p aleatoriamente.
(3) Mientras $t < 3$. Seleccionar dos padres y aplicar las operaciones de intercambio genético y mutación e insertar a los nuevos hijos en la población.
(4) Se utiliza la estrategia elitista en la que el 10% de los mejores de la generación anterior son copiados a la siguiente generación.

Algoritmo 6. ***"Seeding" Approach (modified)***

(1) Se fijan intervalos de t de 3 minutos
(2) Si es el primer intervalo de 3 minutos, inicializar la población aleatoriamente. Sino población = población final del intervalo anterior.
(3) Mientras $t < 3$. Seleccionar dos padres y aplicar las operaciones de intercambio genético y mutación e insertar a los nuevos hijos en la población.
(4) Se utiliza la estrategia elitista en la que el 10% mejores de la generación anterior son copiados a la siguiente generación.

- ***Multi-Objective Neighborhood Search Differential Evolution*** [5]

  Este método propone resolver el problema mediante algoritmos evolutivos, concretamente mediante el *Neigborhood Search Differential Evolution*. El problema se plantea de manera estática (5.1), por lo que se trabaja con un determinado número de aviones y se formula como un problema multi-objetivo (1.2) en el que se minimiza el coste de la secuencia (2.1) y el retraso total de todos los aviones (2.4).

  Tal y como se plantea el problema, está sujeto a 3 restricciones: todos los aviones deben aterrizar dentro de una ventana temporal especificada (3.4), entre aterrizaje de dos aviones consecutivos se debe respetar una distancia de seguridad (3.3) y la relación de precedencia entre aviones que van por la misma ruta (3.5).

  El método plantea resolver el problema en dos fases. La primera fase en la que se busca una única secuencia de aterrizaje que garantice una buena compensación entre los dos objetivos, para la cual se ha diseñado el método *Non-dominated Average Ranking(NAR)* (6.2). La segunda fase en la que la secuencia anteriormente obtenida se utiliza como input en el *Multi-Objective Evolutionary Algorithm (MOEA)* para buscar un conjunto de soluciones (non-dominated), donde se resuelve utilizando una versión multi-objetivo del método *Neigborhood Search Differential Evolution.*

Descripción paso- a- paso del algoritmo:

Notación:

- fr_i , regiones factible de la secuencia de aterrizaje en la que no se viola ninguna restricción, ni de ventanas temporales ni de separación de seguridad entre aviones;
- *TC*, coste total de la secuencia de aterrizaje;
- *STA*, tiempo de aterrizaje programado;
- *ETA*, tiempo estimado de aterrizaje, es el tiempo en el que se estima tendría que aterrizar el avión;
- D_i , es el coste por unidad de tiempo por aterrizar después del ETAi para el avión i;
- A_i , es el coste por unidad de tiempo por aterrizar antes del ETAi para el avión i;
- δ_i, es la coste de penalización para un avión i que aterriza antes o después del ETAi , va en función de D_i y A_i ;
- *OC*, total de *holding loops* de todos los aviones.
- *AC*, es la bondad de la secuencia en el coste total que se estima mediante una media de 20 muestras aleatorias.

Algoritmo 7. ***Non-Dominated Average Ranking***

(1) Enumerar todas las secuencias de aterrizaje posibles basadas en la restricción de precedencia.
(2) Para cada secuencia de aterrizaje posible, calcular las regiones factibles (denotadas por fr_i) de los tiempos estimados de aterrizaje para los aviones en la secuencia, basándose en las restricciones de las ventanas temporales y las separaciones entre aviones.
(3) Para casa secuencia, calcular el total OC. Esta cantidad expresa el total *STA*.
(4) Para cada secuencia, obtener una *x* aleatoriamente escogida de entre todo el conjunto de fr_i , luego el coste total incurrido de esta *x* se calcula mediante:

$$TC = \sum_{i=1}^{N} (STA_i - ETA_i)\delta_i$$

$$where\ \partial_i = \begin{cases} D_i, & STA_i \geq ETA_i \\ A_i, & STA_i < ETA_i \end{cases}$$

(5) Se clasifican las posibles secuencias por OC y AC, respectivamente, con el 1 siendo la mejor clasificación.
(6) Se calcula una puntuación para cada secuencia *no-dominada* sumando las clasificaciones para OC y AC. (Es decir, si una secuencia es la 2ª mejor en cuanto a la clasificación y la 3ª mejor en cuanto a la clasificación AC, su puntuación será 2+3=5). Finalmente la secuencia con la mejor clasificación será utilizada como secuencia de aterrizaje, la cual será optimizada en la segunda fase.

Notación:

- *P* es la población;
- *A* es la secuencia de aterrizaje ;
- *C* es la población de niños;
- *t*, contador de generaciones;
- *NP* ,número de individuos de la población;

Algoritmo 8. ***Multi-Objective Neihgborhood Search Differential Evolution***

Pasos principales:
(1) Fijar $P = \phi,\ A = \phi,\ C = \phi,\ t = 0$.
(2) Inicializar *P* con NP individuos, $P = \{p_1, p_2, \dots, p_{NP}\}$ y establecer $A = P$.
(3) Mientras $t < t_{max}$(i.e. número máximo de generaciones)
(3.1) Generar *NP* individuos mutados $\{v_1, v_2, \dots, v_{NP}\}$ mediante la siguiente ecuación:

$$\mathrm{v_i} = \mathrm{x_{i_1}} + \begin{cases} (\mathrm{x_{i_2}} - \mathrm{x_{i_3}}) \cdot \mathrm{N}(0.5, 0.3)\,, & \text{if } \mathrm{U}(0,1) < 0.5 \\ (\mathrm{x_{i_2}} - \mathrm{x_{i_3}}) \cdot \delta\,, & \text{otherwise} \end{cases}$$

Donde x_{i_1} es seleccionada aleatoriamente del archive, y x_{i_2} y x_{i_3}son aleatoriamente seleccionadas de *P*. N y δ denotan una variable Gaussiana aleatoria y una variable de Cauchy aleatoriamente, respectivamente.

(3.2) Se utiliza el siguiente operador para el intercambio genético para generar *NP* hijos y actualizar *C* con estos nuevos hijos:

$$u_i(j) = \begin{cases} v_i(j), & if\ U_j(0,1) < CR \\ x_i(j), & otherwise \end{cases}$$

(3.3) Elegir a los individuos superiores de *P* y *C* y hacer una nueva población de padres.

For *i*=1 : NP

(a) Si p_i domina c_i, c_i es rechazada.

(b) Si p_i es dominada por c_i , p_i es reemplazada por c_i y se actualiza el archivo A con Archive-Updating*.

(c) Si p_i y c_i son no-dominadas entre ellas, utilizar el Archive-Updating*, para comparar c_i con *A* la menos concurrida con A será la nueva p_i.

(3.4) Interrumpir el archive cuando el tamaño de *A* exceda el máximo.

(3.5) Establecer $t = t + 1$.

(4) Los individuos no-dominados de *A* son las soluciones finales.

*Archive-Updating:

Si c_i es dominado por algún miembro de *A*, descartar c_i

Sino si c_i domina un conjunto de miembros de *A*, eliminar esos miembros de *A* y añadir c_i a *A*

Sino añadir c_i to *A*

- ***Semantic Agents Negotiation Mechanism (Genetic Algorithm and Negotiation Rules)*** **[***6***]**

Este método propone resolver el problema mediante *Semantic Agent Negotiation Mechanism* basado en un sistema Multi-Agente. El problema se plantea de manera estática (5.1), por lo que se trabaja con un determinado número de aviones.

El problema se formula como un problema multi-objetivo (1.2) en el que se minimiza el coste de la secuencia (2.1) y la desviación de los aviones (2.4).

Esta propuesta utiliza un algoritmo genético para establecer la secuencia de caída, y mediante las estrategias de *Proposal Generation Mechanism* (4.2) y *Negotiation Strategies* (4.3) optimizan esta secuencia.

Descripción paso- a- paso del algoritmo:

Notación:

- *S* es la *Negotiation Strategies*, estrategia de negociación;
- *I* es el *Proposal Generation Mechanism*, mecanismo de generación de propuestas;
- *GA* es el Algoritmo genético ;
- *A i* : agente avión, i ∈ {1,…,N}
- *Ag* ={A_1, …, A_n, C} , es el conjunto de agentes avión y el agente control
- *V* = <v_1^p,…, v_n^p , v_c^p > : v_i^p es la valoración del agente avión *i* para propuesta *p*.
- *R* es la secuencia de aterrizaje
- *M*=<Ag, GA, S, I, V, R> : tupla del modelo

Algoritmo 9. ***Semantic Agent Negotiation Mechanism***

(1) Ai utiliza el GA para obtener una secuencia.
(2) Ai envía su secuencia a otros agentes avión y al agente control
(3) Los agentes avión y control puntúan la secuencia según sus propios beneficios. Cuanto menor sea la desviación y el coste, mayor puntuación.
(4) Calcular la nota para cada secuencia. Aquella con la mejor puntuación, será seleccionada como solución inicial.
(5) La solución inicial no es adecuada para todos aviones. Los agentes avión utilizan Proposal Generation Mechanism I para crear una propuesta para la solución inicial.
(6) Todos los aviones utilizan la Negotiation Strategies S para decidir qué propuesta será aceptada. V será utilizada en este paso.
(7) Si las condiciones (ventanas temporales o suspensión de la negociación) no se satisfacen, volver a (5), sino guardar la secuencia en R, y devolver R.

- ***Dynamic programming on a Network Model*** [*7*]

Este artículo, plantea el problema ALSP como una modificación del *Shortest path on a Network* (7.4) y lo resuelve mediante *Dynamic Programming* (7.3). El método da solución a un problema con una única función objetivo (1.1), que en este caso, se trata de la minimización del *makespan* (2.1). Las restricciones que se contemplan son la separación entre aviones (3.3), las ventanas temporales de cada avión (3.4), la precedencia entre aviones (3.5) y el número máximo de cambio de posiciones que se le puede aplicar a cada avión (3.6).

En este método para establecer la secuencia inicial de permiso de aterrizaje se realiza mediante *Constrained Position Shifting* (6.2), de tal forma que el primer avión en entrar en el horizonte del aeropuerto, es el primero que aterriza, y después se busca una solución óptima mediante el algoritmo descrito a continuación.

Descripción paso- a- paso del algoritmo:

Notación:

- *j* son los nodos de la red
- *p*, son las fases del problema, es decir las *posiciones;*
- *T(s)*, tiempo mínimo en el que la secuencia entera puede comenzar;
- *T(t)*, tiempo mínimo en el que la secuencia puede ser completada*;*
- *l(s)* tiempo máximo en el que la secuencia en el nodo s puede comenzar;
- *P(j)*, conjunto de nodos predecesores de j*;*

Algoritmo 10. Procedimiento para calcular el mínimo *makespan*

(1) Asignar T(s) ← 0, l(s) ← 0;
(2) para cada $p = 1, \dots, n$ hacer
(3)para cada nodo *j* en fase *p*

$$T(j) = \max\{e(j), min_{i \in P(j): t(i) \le l(i)}(T(i) + d_{ij})\}$$

$$pred(j) = \arg min_{i \in P(j): t(i) \le l(i)}(T(i) + d_{ij})$$

$$T(t) = min_{i \in P(j): t(i) \le l(i)} T(i)$$

$$Pred(t) = arg\ min_{i \in P(j): t(i) \le l(i)} T(i)$$

- ***Hybrid meta-heuristic****[8]*

  Este método propone una solución al ALSP basada en un modelo de programación mixta (7.2). El problema se plantea como estático (5.1).

  Trata de optimizar una única función objetivo (1.1), minimizar el coste total de la desviación de los tiempos de aterrizaje de los aviones (2.2) sujeto a las restricciones de separación de seguridad entre aviones (3.3), las ventanas temporales de cada avión (3.4) y la restricción de asignación de pista (3.7) ya que el problema contempla varias pistas de aterrizaje y así se controla que un avión aterriza en una única pista.

  Este método utiliza una heurística de construcción (6.5) para encontrar una primera solución. Luego implementa una heurística *Simulated Annealing(SA)* que en combinación con las heurísticas de *Variable Neighborhood Descendent* (VND) y *Variable Neighborhood Search* (VNS) mejoran la solución inicial (las combinaciones son SA+VND y SA+VNS). La idea principal de estas heurísticas es que un óptimo local relacionado a una posible solución necesariamente no tiene que ser un óptimo local relacionado con otra solución. Para escapar de óptimos locales se cambia la estructura de la solución.

Pasos de la heurística *Simulated Annealing*

Notación:

- *x* , solución inicial;
- *t*, es un parámetro de control correspondiente a la temperatura en la analogía del recocido de acero;

Algoritmo 11. Simulated Annealing

(1) Construir una solución inicial *x* utilizando el Algoritmo de Construcción*
(2) Seleccionar una temperatura inicial $t_0>0$;
(3) Repetir hasta que se cumpla la *condición*:
  (3.1) Aplicar algoritmo VND** / Algoritmo VNS*** para mejorar la solución *x*.
  (3.2) Criterio de aceptación:
    Calcular: $\Delta = f(x') - f(x)$ donde *x´* es la solución mejorada.
    Si $\Delta<0$, *x*:=*x´*;
    Sino si $random(0,1) < e^{\Delta/t}$, entonces *x*:=*x´*;
    Sino no aceptar la nueva solución *x´;*
    Actualizar *t*;

Notación:

- *r*, es el conjunto de pistas con el que se trabaja;
- *m*, número de pistas;
- T_i , tiempo de aterrizaje de los aviones;
- *i*, son los aviones*;*
- *n*, número de aviones a ordenar;
- *S* , secuencia de aterrizaje;

Algoritmo 12. Heurística Construcción*

(1) $r > 1$ (r=1…m)
  i=1,…,n;
(2) Ordenar todos los aviones T_i de manera ascendente, de forma que $S = \{i_1, i_2, \dots, i_{n-1}, i_n\}$ donde

$T_{i_1} \leq T_{i_2} \leq \cdots T_{i_{n-1}} \leq T_{i_n}$

(3) De *S* asignar el primer avión, i.e. i_1a la pista *r*=1;
(4) Para *r*=1:m y para todos los aviones $i \in S$
(4.1) Para cada pareja de aviones i_k y i_{k+1} si $T_{i_k} < T_{i_k} + s_{i_k i_{k+1}}$ asignar al avión i_{k+1} la pista *r*=*r*+1.
Sino asignar a i_k y i_{k+1} la pista r=r. Actualizar *S*.
(5) Realizar paso (4) hasta que $S = \emptyset$

**Pasos algoritmo VND
Notación

- N_k, conjunto de estructuras esntorno (*neihgborhood*);
- k, número de estructuras (*neihgborhood*);
- x, es la solución inicial;
- *x´*, es la posible solución encontrada;

Algoritmo 13. VND**

(1) Seleccionar un conjunto de estructuras entorno $N_k,\ k = 1,2, \dots, k_{max}$
(2) Encontrar una solución inicial *x*
(3) Repetir hasta que no haya mejora
(3.1) Asignar *k:*=1;
(3.2) Repetir hasta que *k*=k_{max}
Exploración del "vecindario": encontrar el mejor vecino *x´* de *x*. ($x' \in N_k(x)$)
Mover o no, si la solución *x´* es mejor que *x*, asignar $x \coloneqq x'$y $k \coloneqq 1$, sino $k \coloneqq k + 1$

Notación:

- N_k, conjunto de estructuras entorno
- k, número de estructuras entorno
- x, es la solución inicial;
- *x´*, es la posible solución encontrada;

Algoritmo 14. VNS***

(1) Seleccionar un conjunto de estructuras entorno $N_k,\ k = 1,2, \dots, k_{max}$ que serán utilizadas para la busqueda;
(2) Encontrar una solución inicial *x*, y una condición de stop
(3) Repetir hasta que se cumpla la condición de stop
(3.1) Asignar *k:*=1;
(3.2) Repetir hasta que *k*=k_{max}
- *Shaking:* generar un punto *x´* aleatorio en el *k-esimo* entorno de $x(x' \in N_k(x))$
- Búsqueda local: aplicar algún método de busquedalocal con *x´*como solucion inicial, denotar el óptimo local como $\bar{x}$.
- Mover o no: siel optimo local es mejorca que el original, moverse ($x \coloneqq \bar{x}$), y continuar la búsqueda con N_1 y *k:*=1; sino asignar *k:*=*k*+1.

4.3. Software

Existe un número de software disponible para dar soporte a la gestión de aviones en el aeropuerto. Estos sistemas incluyen, en parte, módulos para programar los aterrizajes y despegues. Los más utilizados son:

- COMPAS (en Frankfurt). Coge una heurística inicial para determinar la secuencia y aplica un procedimiento de enumeración designado a eliminar los conflictos de los tiempos de separación.
- OASIS utiliza un procedimiento de búsqueda basado en A*.
- CTAS (Dallas Fort Worth). Es el más extendido. Contiene dos módulos dirigidos al problema considerado de secuencia de aterrizaje de aviones.
  a) Módulo de secuencia y programación, el cual:
     i) Secuencia el aterrizaje de aviones en una pista utilizando una heurística constructiva simple basada en la fusión de secuencias parciales
     ii) Programa utilizando la siguiente expresión: STA=max[STA of previous plane + separation time, earliest posible landing time]
  b) Módulo de asignación de pista

4.4. Conclusiones

Aunque muchos artículos de investigación sobre el ALSP se han publicado durante las últimas tres décadas, no se han desarrollado métodos que hayan sido implementados. Las razones pueden residir en que los métodos puedan suavizar o descartar restricciones operacionales críticas, que los algoritmos necesiten tiempos de ejecución poco razonables, que estudien entornos estáticos en lugar de entornos dinámicos, que se ignoren los requisitos de varios *stakeholders*, o dependan de características de un aeropuerto especifico. Estudios existentes generalmente consideran algunas de las restricciones más comunes y obvias, mientras ignoran las restricciones operacionales que se pueden observar del trabajo diario de los controladores.

Por consiguiente, se espera que se puedan encontrar soluciones para que puedan ser usadas por los controladores aéreos. Ya que el problema es complejo (es un problema *NP-hard*) y requiere de requisitos de varios *stakeholders*, los métodos con heurísticas multi-objetivo parecen ser más apropiados que no aquellos métodos con un único objetivo, como la programación dinámica, que suelen resultar más exigentes desde un punto de vista computacional.

5. Análisis de los métodos de optimización Multi-objetivo existentes

5.1. Optimización Multi-Objetivo

El proceso de optimizar simultáneamente un conjunto de funciones objetivo se llama Optimización Multi-Objetivo. Por lo general un problema Multi-Objetivo se define como:

K	número de funciones objetivo
J	número de restricciones desigualdad
L	número de restricciones igualdad
N	número de variables de decisión x_i
g	vector de restricciones desigualdad
h	vector de restricciones igualdad
$x \in X$	vector de variables de decisión
F	vector de funciones objetivo

$$Minimize\ F(x) = [F_1(x), F_2, \dots, F_K(x)]^T$$

$$subject\ to\ g_j(x) \leq 0, j = 1,2 \dots J$$

$$h_l(x) = 0, \qquad l = 1,2 \dots L$$

Generalmente, no hay una única solución global óptima para este tipo de problemas, sino un conjunto de soluciones alternativas. Estas soluciones son óptimas en el sentido de que en el espacio de búsqueda no hay mejores soluciones cuando se consideran todos los objetivos. Se las conoce como soluciones Pareto-óptimas.

Un punto $x^* \in X$ es Pareto-óptimo si no existe otro punto, $x \in X$,de manera que $F(x) \leq F(x^*), y\ F_i(x) < F_i(x^*)$ para almenos una función. Es decir, un punto es Pareto-óptimo si no hay otro punto que mejore al menos una función objetivo sin detrimento de otra función objetivo.

En optimización Multi-Objetivo se utiliza el concepto de dominación. Una solución $x^{(1)}$se dice que domina a otra solución $x^{(2)}$ si se cumple que:

(1) La solución $x^{(1)}$ no es peor que $x^{(2)}$ en todos los objetivos, esto es,
$F_i(x^{(1)}) \trianglerighteq F_i(x^{(2)}) \quad \forall\ i = 1,2, \dots, K$

(2) La solución $x^{(1)}$ es estrictamente mejor que $x^{(2)}$ en al menos un objetivo, esto es,
$F_{\bar{\imath}}(x^{(1)}) \lhd F_{\bar{\imath}}(x^{(2)}) \quad para\ al\ menos\ una\ \bar{\imath} \in \{1,2, \dots, K\}$

Donde el operador $\lhd$ se utiliza entre dos soluciones i y j como $i \lhd j$ para denotar que la solución i es mejor que la j en un objetivo particular, y de forma similar $i \rhd j$ implica que la solución i es peor que la solución j en dicho objetivo.

Existen tres relaciones de dominancia posibles entre dos soluciones $x^{(1)}$ y $x^{(2)}$: (i) la solución $x^{(1)}$ domina a $x^{(2)}$, (ii) $x^{(1)}$ es dominada por $x^{(2)}$, (iii) $x^{(1)}$ y $x^{(2)}$ no se dominan entre ellas.

Para un conjunto finito de soluciones dado *P*, si se realizan todas las comparaciones posibles entre soluciones emparejadas se obtienen cuales soluciones dominan a cuales soluciones y cuales soluciones son no-dominadas entre ellas. Al final, se obtiene un conjunto *P'* de soluciones las cuales no son dominadas entre sí. Este conjunto tiene la propiedad de dominar a todas las demás soluciones que no pertenecen a dicho conjunto, es decir, las soluciones de este conjunto son las mejores comparadas al resto de soluciones. Este conjunto *P'* se conoce como el conjunto Pareto-óptimo.

Formatat: Tipus de lletra: Cursiva

5.2. Características de los métodos de optimización Multi-objetivos

Los métodos capaces de resolver problemas de optimización multi-objetivo se representan mediante algoritmos multi-objetivos evolutivos (MOEAs) que presentan las siguientes ventajas[[9]:

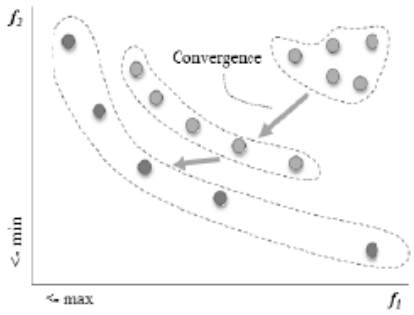
a) evolucionar un conjunto de soluciones no dominadas en lugar de una única solución;
b) independencia ante un problema con una estructura fuerte, como convexidad y discontinuidad de las funciones objetivo;
c) habilidad para buscar un espacio de gran dimensión con diferentes rangos de variables de decisión;
d) habilidad para codificar variables de decisión discretas y continuas;
e) compatibilidad con técnicas de manejo de restricciones;

5.2.1. Técnicas para preservar la diversidad en el frente Pareto óptimo

El problema de la convergencia se basa en buscar maneras de mejorar la precisión de la aproximación del frente Pareto-óptimo y mantener su diversidad (Gráfica 5.1.). Esto puede ser resuelto con una clasificación basada en la asignación de bondad que requiere el uso de diferentes métricas como:

a) Dominancia en profundidad que implica dividir la población en diferentes frentes para reflejar un frente de una solución determinada.
b) Dominancia en ranking, calculada como el número de soluciones en las cuales una determinada solución es dominada.

c) Dominancia en contador que se calcula de acuerdo al número de soluciones que domina cierta solución.

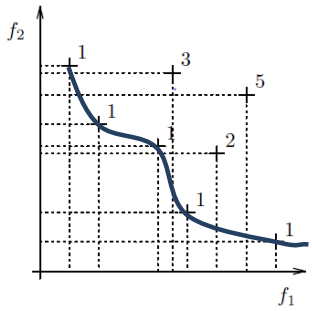


Gráfica 5.1. Convergencia del algoritmo del frente Pareto-óptimo

Se considera un individuo x_i de la generación t dominado por $p_i^{(t)}$. Su posición en el ranking de individuos viene dada por:

$$rank\ (x_i, t) = 1 + p_i^{(t)}$$

A todos los individuos no-dominados se les asigna rank 1 (Gráfica 5.2). Establece que el individuo etiquetado con el 3 en la figura es peor que el individuo etiquetado con el 2, ya que este último se encuentra en una región en la que la compensación entre funciones objetivos está peor descrita por los individuos restantes.



Gráfica 5.2 Clasificación soluciones Multi-Objetivo

La dominancia en profundidad, permite evitar la deriva genética del algoritmo, el cambio en la frecuencia relativa con la cual los alelos ocurren en una población, que aparece debido a errores estocásticos en el proceso de selección. La convergencia del algoritmo al frente Pareto-óptimo obtenido como resultado de aplicar dicho procedimiento se muestra en la Gráfica 5.2.

Los procesos de dominancia en ranking y profundidad permiten al algoritmo tratar de igual forma todas las soluciones no-dominadas, lo que conlleva a una mejor convergencia.

5.2.2. Clasificación basada en asignación de bondad para mejorar la convergencia de los algoritmos.

La cuestión de la preservación de la diversidad está estrechamente relacionada con la incorporación de información de la densidad en el proceso de selección. Las técnicas pueden ser clasificadas de acuerdo a las categorías de técnicas estimación de densidad estadísticas.

Fitness sharing. Esta técnica promociona las soluciones de las regiones menos pobladas en el espacio de búsqueda (Gráfica 5.3.). Define el entorno de la solución *i* en términos de la *sharing function* $sh(d_{ij})$, la cual toma la distancia d_{ij} a otra solución *j* como argumento. En la práctica, el tamaño del entorno llamado nicho se controla por la distancia σ_{share}.

Al principio, el recuento de soluciones del nicho localizadas en el mismo nicho se calcula como la suma de los valores de las *sharing functions*. Representa la densidad estimada para la solución *i*. Después, el *shared fitness* ψ_i de la solución *i* se calcula como el *fitness dummy* ψ'_i obtenido en el proceso de clasificación y disminuido proporcionalmente al recuento de soluciones del nicho que comparten en el mismo entorno:

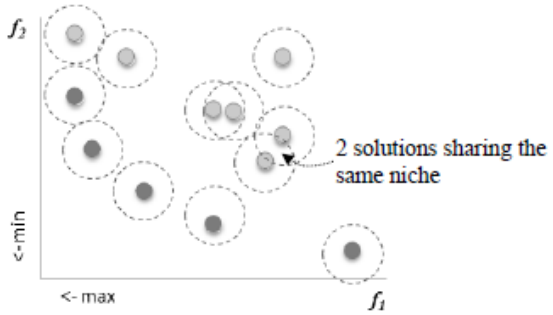
$$\psi_i = \frac{\psi'_i}{\sum_{j \in i} sh(d_{ij})}$$

Donde la distancia normalizada entre dos soluciones d_{ij} se mide utilizando el genotipo (distancia Hamming) o fenotipo (distancia Euclidiana) de cualquiera de las dos soluciones, y la *sharing function* $sh(d_{ij})$:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha}, & si\ d_{ij} < \sigma_{share} \\ 0, & caso\ contrario \end{cases}$$

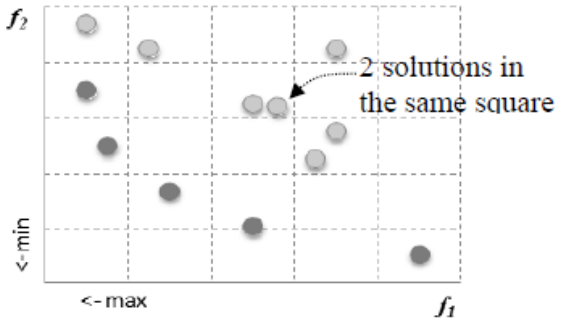
Donde α es un factor de ajuste.

Los inconvenientes de esta técnica incluyen la necesidad de especificar el tamaño del nicho σ_{share}, así como el tiempo requerido para computar los valores de *shared fitness*.



Gráfica 5.3. Representación gráfica de la técnica de fitness sharing

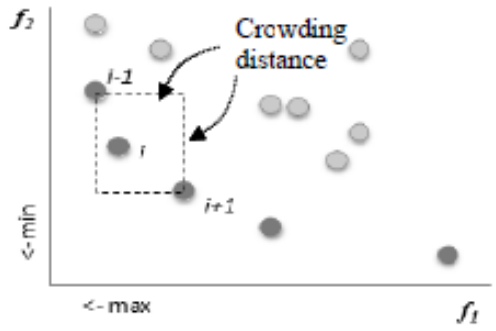
Grid-base niching. Esta técnica se basa en dividir el espacio objetivo en cuadrados y evaluar la densidad alrededor de una determinada solución contando el número de soluciones en el mismo cuadro de la cuadricula (Gráfica 5.4). Adicionalmente, el tamaño del nicho σ_{share} se utiliza para determinar el tamaño del cuadro. Esta característica hace que esta técnica sea similar a la técnica de *fitness sharing.*



Gráfica 5.4. Representación gráfica de la técnica *Grid-base niching*

Clustering. Esta técnica categoriza las soluciones mediante las medias de la medida de las distancias Euclidianas entre ellas. Primero, a cada solución de una población se le asigna a un clúster separado. Después, se calcula la distancia Euclidiana entre todos los pares de soluciones. Finalmente, los clúster con las menores distancias se fusionan para formar un clúster mayor. Se ha probado que esta técnica puede producir una amplia diversidad de frentes Pareto-óptimos, pero requiere un tiempo de ejecución adicional para formar los clústeres en cada iteración del algoritmo.

Crowding. Esta técnica mide la distancia normalizada entre dos soluciones cercanas a cada lado de la solución a lo largo de cada uno de las funciones objetivo. Estas distancias se resumen en el resultado de distancia *crowding* para una solución. Por lo tanto, una densidad se representa como la aglomeración de una región alrededor de una determinada solución (Gráfica 5.5). Si una solución proporciona el mejor valor conocida para cualquier función objetivo, entonces su distancia *crowding* se establecerá como infinito. Las soluciones con mayores distancias *crowding* indican regiones menos aglomeradas. Esta técnica es computacionalmente más rápida que las técnicas de *fitness sharing* y *clustering*, pero la diversidad del frente Pareto-óptimo por lo general es peor.



Gráfica 5.5. Representación gráfica de la técnica crowding

k-th nearest neighbour method. Este método representa la densidad como una función de cambio de distancias entre soluciones determinada entre sus vecinos más cercanos. De esta forma, las distancias de una solución determinada a otras soluciones de definen en el espacio objetivo, y se ordenan en orden descendente. El primer individuo de la lista contiene la distancia a la solución más cercana. Con el objetivo de mantener la diversidad del frente Pareto-óptimo, el valor *k* es predefinido. De esta forma, la densidad de una determinada solución se calcula como la inversa de la distancia *k-ésima* más un valor reciproco.

5.2.3. Influencia de los mecanismos de selección en la convergencia de los algoritmos

El mecanismo de selección en un algoritmo evolutivo multi-objetivo influye en el número de selecciones para emparejamiento y en el reemplazo para cada solución de acuerdo a su *fitness*. De hecho, controla la convergencia del algoritmo en el frente Pareto-óptimo. Generalmente, la selección se lleva a cabo en dos pasos, primero todas las soluciones se clasifican basándose en la relación de dominancia y luego esta clasificación se refina utilizando la técnica de nicho.

Normalmente, los mecanismos de selección que se implementan son:

Selección *fitness-proportionate.* Este mecanismo utiliza el muestreo de la "rueda de la ruleta" de manera que el *fitness* de cada cromosoma es proporcional a una porción de la "rueda de la ruleta", esto se traduce en que los cromosomas más aptos obtienen una sección mayor a aquellos que son menos aptos. Para seleccionar los cromosomas con mejor *fitness* para cruzar, la rueda se gira N veces, donde N es el tamaño de la población. En cada giro, se elige al individuo que tenga la sección en la que la ruleta se ha parado. Este mecanismo se tiende a combinar con la técnica de *fitness sharing.* Este tipo de selección causa una prematura convergencia debido a que la presión selectiva de los mejores cromosomas de una determinada población es más alta que la presión de peores cromosomas. Esto da como resultado la búsqueda de un frente Pareto-óptimo en una región local y puede llevar al estancamiento en la región que está lejos del verdadero frente.

Selección *binary tournament.* Para seleccionar un progenitor se eligen de forma aleatoria dos soluciones de la población. Una vez obtenidas las soluciones, se genera un número aleatorio en el intervalo [0,1] para compararlo con un parámetro límite. Para seleccionar un progenitor para cruzar, se comparan los factores de las soluciones obtenidas de forma aleatoria, la comparación se realiza respecto al *fitness* y la diversidad del entorno *crowding.* Aquella solución que tenga un mejor *fitness* se selecciona como progenitor, en el caso en el que el *fitness* sea igual para ambas soluciones, se compara el factor de diversidad del entorno. La selección *binary tournament* proporciona una mejor convergencia que la selección *fitness-proportionate*.

5.2.4. Mecanismos elitistas para conservar soluciones no dominadas

Los mecanismos elitistas se utilizan para prevenir que se pierdan las soluciones no-dominadas. La forma en la cual se guardan las soluciones de una iteración a otra del algoritmo evolutivo se define mediante estrategias elitistas. Las estrategias utilizadas más comunes son:

Estrategia de *Selección* $(\mu + \lambda)$ que combina poblaciones de padres de entre el *mating pool* para seleccionar la mejor combinación de soluciones de la población mixta. La desventaja de esta estrategia es que las soluciones no-dominadas nuevas no se pueden adaptar a la siguiente población en el caso que las soluciones hijo son dominadas por las soluciones padre.

Estrategia de *almacenamiento externo* donde las soluciones no-dominadas encontradas a lo largo del proceso de optimización se almacenan es una población secundaria llamada archivo externo para mejorar la distribución del frente Pareto-óptimo. El archivo externo se actualiza con nuevas soluciones que dominan las actuales. La gran desventaja de este mecanismo elitista es el continuo crecimiento de externo archivo. De hecho, puede reducir la presión selectiva y ralentizar la búsqueda, ya que las soluciones no-dominadas participan en el proceso de selección. Para enfrentarse a esta dificultad, los miembros del archivo se pueden comparar

respecto a la concurrencia de su entorno y aquellos localizados en un entorno más concurrido pueden ser eliminados del archivo externo.

5.3. Análisis de los algoritmos Multi-Objetivos

Los Algoritmos Evolutivos Multi-Objetivos (Tabla 5.1) comparten propiedades de adaptación mediante un proceso iterativo que acumula y amplifica el beneficio mediante prueba y error.

Técnicas		Algoritmos Evolutivos Multi-Objetivo						
		MOGA	NSGA	NPGA	SPEA	SPEA-II	NSGA-II	PAES
Mecanismos de asignación de *fitness* basados en el ranking	Dominancia profundidad		x				x	
	Dominancia contador				x	x		
	Dominancia ranking	x		x	x	x		x
Mecanismos de preservación de la diversidad	Fitness sharing	x	x					
	Grid-based niching			x				x
	Clustering				x			
	Crowding						x	
	k-th nearest neighbour method					x		
Mecanismo de selección	Fitness-proportionate	x	x					
	Binary tournament			x	x	x	x	x
Mecanismos de elitismo	No-elitista	x	x	x				
	Almacenamiento externo				x	x		x
	(μ+λ)						x	

Tabla 5.1 Resumen de los algoritmos evolutivos más conocidos

Las soluciones candidatas representan miembros de una población virtual que intentan sobrevivir en un entorno definido por las funciones objetivo específicas. En cada caso, el proceso evolutivo refina la capacidad adaptativa de la población de las soluciones candidatas en el entorno, generalmente utilizando representaciones de mecanismos evolutivos como combinación genética y mutación.

5.3.1. Multi-Objective Genetic Algorithm (MOGA)

El algoritmo MOGA [10] fue uno de los primeros algoritmos evolutivos multi-objetivos introducido por Fonseca y Fleming. Este algoritmo es bastante similar al algoritmo genético debido a la selección de *fitness-proportinate*, un punto de *crossover* único y una mutación *bit-wise*.

En el MOGA se compara la población entera. A los individuos no-dominados se les asigna *Rank* =1 y a los demás individuos se les asigna un valor de *Rank* comparando la no-dominancia entre ellos de la siguiente manera: para un individuo se busca el número de individuos que lo

dominan de forma estricta. Luego, al final del proceso de clasificación, podría haber un número de puntos que tengan en mismo *Rank*, entonces el procedimiento de selección utiliza estos rankings para seleccionar o eliminar estos puntos del *mating pool*.

5.3.2. Non-Dominated Sorting Genetic Algorithm (NSGA)

El objetivo del algoritmo NSGA [11] es mejorar la capacidad de adaptabilidad de la población de las soluciones candidatas al frente Pareto limitadas por un conjunto de funciones objetivo. Asigna valores *fitness* respecto al frente no-dominado de una solución particular utilizando un procedimiento de asignación de *fitness* basado en la dominancia en profundidad.

Antes de que se lleve a cabo la selección, la población es clasificada en base a la no-dominación de los individuos. Primero se identifican los individuos no-dominados presentes en la población. A continuación, todos estos individuos pasan a constituir el primer frente no-dominado de la población y se le asigna un valor de *fitness dummy* elevado, este mismo valor es asignado para que el potencial de todos los individuos no-dominados sea equitativo. Con tal de mantener la diversidad de la población a los individuos se les aplica un método de *sharing*. Estos métodos de *sharing* realizan una selección utilizando valores de *fitness* degradados que se obtienen dividiendo el valor original de *fitness* de un individuo entre la cantidad proporcional al número de individuos a su alrededor. Después estos individuos no-dominados son ignorados temporalmente para procesar el resto de la población de la misma manera para identificar los individuos para un segundo frente Pareto-óptimo. A este nuevo conjunto de puntos se les asigna un valor de *fitness dummy* inferior al valor del *fitness* del método s*haring* del frente anterior. Este proceso se continúa hasta que la población entera es clasificada en frentes diferentes.

Esta población es reproducida de acuerdo a los valores de *fitness dummy*. Los individuos del primer frente, al tener el valor de *fitness* máximo, siempre tendrán más copias que el resto de poblaciones. Con esto se pretende buscar regiones no-dominadas por frentes Pareto-óptimos. Esto resulta en una rápida convergencia de la población hacia regiones no-dominadas y el *sharing* ayuda a que se distribuya sobre esta región.

5.3.3. Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

El NSGA-II [12] es una versión del NSGA. El NSGA-II introduce mejoras en aquellos puntos débiles del NGSA que son:

1) Complejidad computacional elevada de la ordenación de las soluciones no-dominadas. El algoritmo de ordenación no-dominado en el caso de poblaciones grandes es muy caro dado que la población necesita ser ordenada en cada iteración.

2) Falta de elitismo. Aplicar técnicas elitistas podría acelerar el rendimiento del algoritmo genético de manera significante además de ayudar a prevenir la perdida de soluciones buenas una vez estas han sido encontradas.
3) Necesidad de especificar el parámetro de sharing σ_{share}. Los mecanismos tradicionales para garantizar la diversidad en una población y obtener una amplia variedad de soluciones equivalentes dependen del concepto sharing. El mayor problema es que requiere la especificación del parámetro σ_{share} y aunque se ha trabajado en la dinamización del tamaño de este parámetro, es preferible un mecanismo para la preservación de la diversidad sin parámetros.

Con tal de ordenar una población de tamaño *N* de acuerdo al nivel de no-dominación, las soluciones de una misma población se comparan entre sí para identificar las soluciones no-dominadas. En este punto, se identifican los individuos del primer frente no-dominado, estas soluciones son temporalmente ignoradas para encontrar el segundo frente, y así sucesivamente. Primero para cada solución, se calculan dos identidades: (i) n_i, número de soluciones que dominan a la solución *i;* (ii) S_i, conjunto de soluciones en la que la solución *i* domina y se identifican aquellos puntos que tengan $n_i = 0$ y se incluyen en una lista F_1, que representa el frente actual. Para cada solución en el frente actual se visita cada miembro *j* en el conjunto S_i y se reduce su n_j una unidad. Si al hacer esto, algún miembro j $n_j = 0$ se añade a una nueva lista *H*. Cuando todos los miembros del actual frente han sido comparados, se declara a los miembros de la lista F_1 como miembros del primer frente y se continúa el proceso con el nuevo frente *H* como frente actual.

Para obtener una estimación de la densidad de las soluciones del entorno de un determinado punto de la población se coge la distancia media entre los dos puntos en cualquier lado del punto a lo largo de los objetivos. Esta cantidad $i_{distance}$ sirve como un estimador del tamaño del hipercubo más grande que cerca el punto *i* sin incluir ningún otro punto de la población.

El procedimiento de comparación de la concurrencia se utiliza para garantizar la diversidad de las soluciones no-dominadas se El operador para comparar la concurrencia $\geq_n$ guía el proceso de selección en las diferentes fases del algoritmo hacia un frente Pareto-óptimo uniformemente extendido. Se asume que cada individuo i en la población tiene dos atributos: (1) clasificación de no-dominancia i_{rank} y (2) distancia local de concurrencia $i_{distance}$ con los que se define un orden parcial tal que:

$$i \geq_n j \quad si\ (i_{rank} < j_{rank}) \quad o\ si\ i_{rank} = j_{rank}\ y \quad i_{distance} > j_{distance}$$

Esto es, entre dos soluciones con diferente valor de ranking se prefiere el de menor valor. En caso contrario, si ambos puntos pertenecen al mismo frente, se prefiere aquel situado en la región menos concurrida.

5.3.4. Niched-Pareto Genetic Algorithm (NPGA)

En el NPGA [13] dos candidatos para la selección son elegidos de forma aleatoria de la población y un conjunto de individuos también se elige de forma aleatoria para comparar la dominancia de los individuos. Los candidatos se comparan con cada individuo del conjunto. Si un candidato es dominado por el conjunto de comparación, y el otro no, el último es seleccionado para la reproducción. Si ninguno de los dos son dominados por el conjunto de comparación se implementa la técnica Grid-based niching.

Este algoritmo utiliza el mecanismo de *fitness sharing* para distribuir la población a través de un número determinado de cimas (*peaks*), en el que cada región recibirá una fracción de la población dependiendo de la altura de la cima (*peak's height*). Para obtener esta distribución, *sharing* pronostica la degradación del *fitness* individual de un objetivo F_i por el recuento de nichos (*niche count*) m_i calculada para un individuo. Esta degradación se obtiene dividiendo el *fitness* de un objetivo entre el recuento de nichos para encontrar el *shared fitness*:

$$Sh_i = \frac{F_i}{m_i}$$

El recuento de nichos m_i es una estimación de la concurrencia del entorno (nicho) de un individuo *i*. Esta se calcula para todos los individuos de la población:

$$m_i = \sum_{j \in P} Sh[d[i,j]]$$

Donde $d[i,j]$ es la distancia entre dos individuos y $Sh[d]$ es la función *sharing*. Normalmente se utiliza la función *sharing* triangular donde:

$$Sh[d] = \begin{cases} 1 - \dfrac{d}{\sigma_{share}}, & d \leq \sigma_{share} \\ 0, & d > \sigma_{share} \end{cases}$$

Aquí σ_{share} es el radio del nicho, estimación de la separación mínima deseada o esperada entre soluciones ganadoras fijadas por el usuario. Individuos con una distancia menor a σ_{share} se degradan el *fitness* entre sí ya que se encuentran en el mismo nicho. De esta forma la convergencia ocurre dentro de un mismo nicho, pero se evita la convergencia de la población entera.

5.3.5. Strength Pareto Evolutionary Algorithm (SPEA)

El objetivo del SPEA [14] este algoritmo es localizar y mantener un frente de soluciones no dominadas, en el mejor de los casos un conjunto de soluciones Pareto óptimas. Esto se consigue utilizando un proceso evolutivo para explorar el espacio de búsqueda, y un proceso de selección que utiliza una combinación del grado en el que una solución candidata es dominada y una estimación de densidad del frente Pareto como una asignación de *fitness*. Un archivo del conjunto no dominado se mantiene separado de la población de soluciones candidatas utilizadas en el proceso evolutivo, otorgando una especie de elitismo.

El algoritmo SPEA utiliza una población regular y un archivo (conjunto externo). Empieza con una población inicial y un archivo vacío. Primero, todos los individuos no-dominados de la población se copian al archivo y los individuos dominados o duplicados se eliminan del archivo durante esta actualización del archivo. Si el tamaño del archivo actualizado excede un límite predefinido, se eliminan miembros del archivo utilizando una técnica de *clustering* que mantiene las características del frente no-dominado. Después, a los individuos de la población y del archivo se les asigna un valor de *fitness* ψ_i a cada solucion *i* del archivo externo y su fuerza st_i:

$$st_i = \frac{n_i}{N + 1'}$$

Donde n_i es el número de soluciones dominadas por *i* en la población *P* y *N* su tamaño. En el siguiente paso, se evalúa el *fitness* ψ_i para cualquier solución *j* en la población *P* como la suma de las fuerzas en el archivo externo (ExtA) que débilmente dominan *j*:

$$\psi_i = 1 + \sum_{i \in ExtA\,, i \preccurlyeq j} st_i$$

La fase de selección para el cruce de la unión de la población y el archivo se selecciones mediante *binary-tournaments*. Por último, después de la combinación y mutación la población antigua es reemplazada por la nueva población hija resultante.

Hay algunos puntos débiles en este algoritmo. En la asignación de *fitness* los individuos que son dominados por los mismos miembros tiene un valor de *fitness* idéntico. Esto implica que en el caso en el que archivo contiene un único individuo, todos los miembros de la población tendrán la misma clasificación independientemente de si se dominan entre ellos o no. Como consecuencia, la presión selectiva disminuye sustancialmente y en este caso en particular el SPEA se comporta como un algoritmo de búsqueda aleatorio. Otro punto débil es la estimación de densidad, si varios individuos de la generación en curso son indiferentes, es decir no se dominan entre ellos, se puede obtener muy poca o no obtener información acerca del orden

parcial de las soluciones o en cuento a la relación de dominancia. Por último, el truncamiento del archivo, ya que aunque la técnica de *clustering* es capaz de reducir el conjunto no-dominado sin destruir sus características, puede perder soluciones externas.

5.3.6. Strength Pareto Evolutionary Algorithm II (SPEA-II)

El algoritmo SPEA constituye la base para el algoritmo SPEA-II [15]. Este último fue diseñado para corregir los problemas del SPEA anteriormente mencionados. En contraste con el SPEA, el SPEA-II utiliza una estrategia de asignación de *fitness* que incorpora información de densidad. Además, el tamaño del archivo es fijo, es decir el archivo se completa con individuos no-dominados. En cambio en el SPEA el tamaño del archivo varía con el tiempo. Además, la técnica *clustering* que se utiliza cuando el frente no-dominado excede el límite del archivo ha sido reemplazada por un método alternativo que tiene características similares.

En el SPEA-II el valor de la fuerza se estima para la población y las soluciones del archivo externo. Luego, se calcula el *fitness dummy* como la suma de las fuerzas de las soluciones que dominan a la actual:

$$\psi_i = 1 + \sum_{i \in ExtA \cup P, i \preccurlyeq j} st_i$$

A cada individuo i del archivo $\overline{P_t}$ y la población P_t se le asigna un valor de fuerza st_i, que respresenta el número de soluciones que dominan:

$$st_i = |\{j | j \in P_t + \bar{P}_t \wedge i \succ j\}|$$

Finalmente, añadiendo la densidad *D(i)* al *fitness* aparente (*raw fitness*) *R(i)* de la solución *i* se obtiene el *fitness* de *i*.

$$R(i) = \sum_{i \in P_t + \overline{P_t},\ j > i} st_j$$

$$fitness = R(i) + D()$$

La estimación de densidad se basa en el método del *k-th nearest neighbour method*, donde la densidad en cualquier punto es una función de la distancia a la k-ésima solución más cercana:

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

Donde $k = \sqrt{N + \overline{N}}$, que es el tamaño de la muestra y σ_i^k es la distancia a la k-ésima solución.

5.3.7. Pareto-Archived Evolutionary Strategy (PAES)

Inicialmente PAES[16] se desarrolló como un método de búsqueda local para encontrar soluciones problema al del área de enrutamiento de telecomunicaciones. Más adelante se desarrolló para comprobar si podía encontrar soluciones para un planteamiento multi-objetivo del mismo problema.

Este algoritmo lo componen tres partes: el generador de soluciones candidatas, la función de aceptación de soluciones candidatas y el archivo de soluciones no-dominadas. El generador de soluciones candidatas es similar a una mutación aleatoria mediante *hillclimbing*, este mantiene una única solución y en cada iteración produce un único nuevo candidato mediante mutación aleatoria. Ya que el objetivo de la búsqueda multi-objetivo es encontrar una distribución/conjunto de soluciones, PAES necesita proporcionar un archivo de soluciones no dominadas para mantener un número limitado de soluciones cuando éstas son encontradas por el *hillclimber*. Este algoritmo para elegir entre una mutación y la última solución actual, utiliza el archivo, que proporciona una fuente natural y conveniente de la cual obtener conjuntos de comparativos.

El PEAS utiliza un mecanismo de preservación de la diversidad basado en la división recursiva del espacio objetivo d-dimensional, el mecanismo de *grid-based niching*. Cuando cada solución es generada, se define su posición en la "cuadricula" del espacio objetivo. Asumiendo que el rango de espacio se define en cada objetivo, la localización requerida puede ser encontrada bisecando el rango de cada objetivo y buscar en que mitad reside la solución. Esta recursiva subdivisión del espacio y asignación de localización se lleva a cabo mediante un método adaptativo que elimina la necesidad de un parámetro para el tamaño del nicho. Este método, calcula el rango del espacio objetivo de las soluciones actuales en el archivo y ajustando la "cuadricula" para que cubra todo el rango y luego las localizaciones de cada cuadrícula son recalculadas.

En este algoritmo no es necesaria una fase de selección ya que únicamente hay una solución actual, por lo tanto toda la complejidad reside en la aceptación/rechazo de la solución mutada y en la actualización del archivo.

5.4. Conclusiones

De todos los métodos analizados en esta apartado, se han seleccionado los algoritmos NSGA y NSGA-II para lleva a cabo la demostración y posterior análisis de las capacidades técnicas de los algoritmos multi-objetivo evolutivos (MOEAs). Particularmente se ha seleccionado el NSGA.II ya que en la mayoría de los casos este algoritmo es capaz de encontrar un conjunto se soluciones y una mejor convergencia mejores de la solución aproximada del verdadero frente

Pareto-optimo comparado con los algoritmos PAES y SPEA. El NSGA también se ha seleccionado como primera versión del NSGA y así poder analizar las mejoras y diferencias entre ambas versiones.

6. Desarrollo de métodos multi-objetivo de secuencia de aterrizaje

En esta sección se han desarrollado dos métodos, el NSGA y el NSGA-II explicados en la sección 5. De cada algoritmo se presenta su diagrama de flujo, el pseudo-código propuesto y un ejemplo del funcionamiento del algoritmo.

6.1. NSGA

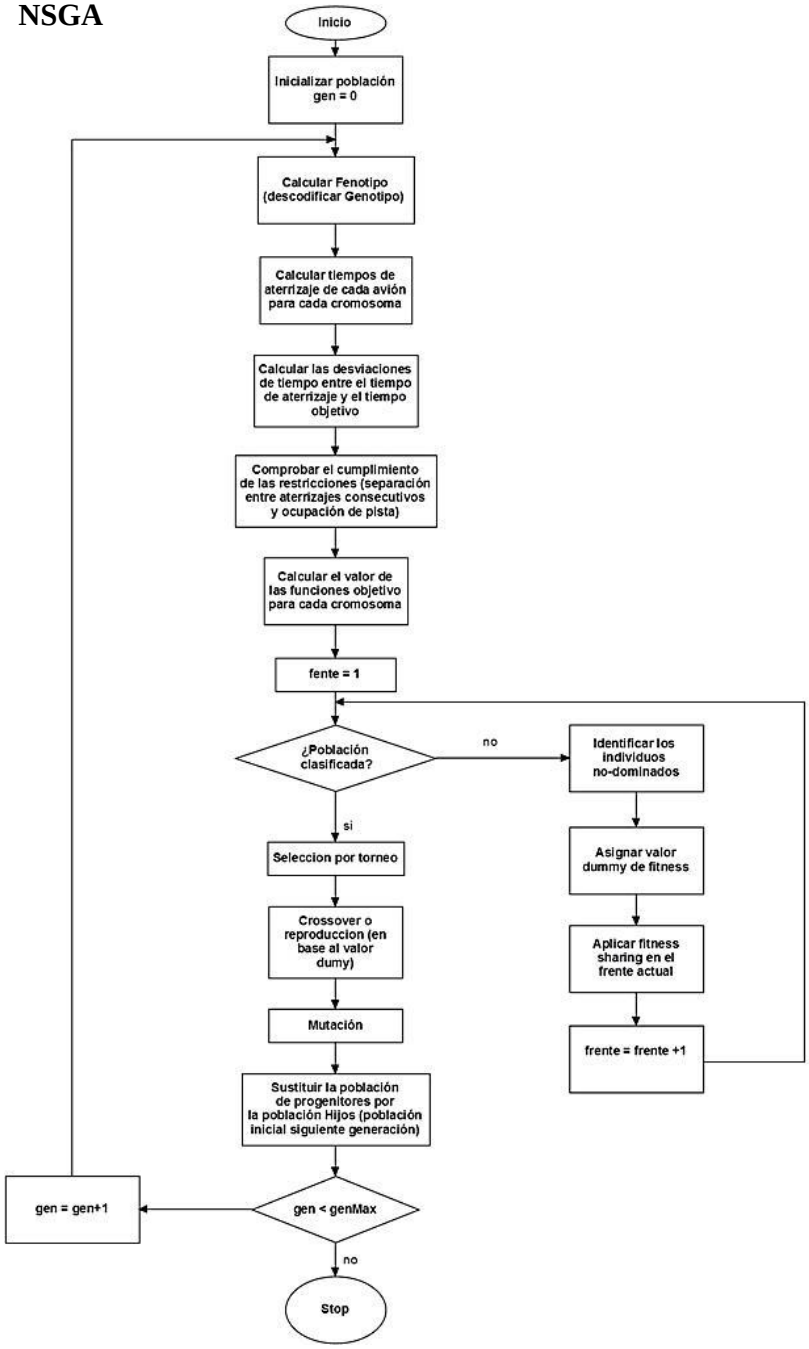


Ilustración 6.1. Diagrama de flujo NSGA

6.1.1. Pseudo-código NSGA

Nomenclatura

K	número de funciones objetivo
np	tamaño de la población
P_0	población inicial
P_g	población de la generación g
H	Hijos
Pr_i	progenitores de ínice i
Pr_op	conjunto de individuos seleccionados para elegir progenitores
F_i	frentes de soluciones no-dominadas $i = 1 \dots n$ donde n es el número de frentes no-dominados de la generación
f_x	*fitness* de una solución
z_k	función objetivo
z_k^{max}	valor máximo de la funcion objetivo z_k encontrada hasta el momento durante la búsqueda
z_k^{min}	valor mínimo de la funcio objetivo z_k encontrada hasta el momento durante la búsqueda
d_{xy}	distancia entre dos individuos en un mismo frente
σ_{share}	distancia máxima permitida entre dos miembros para que pertenenzcan al mismo nicho
g	número de generación
g_{max}	número máximo de generaciones
nr	número generado aleatoriamente que sirve para realizar la reproducción. El número pertenece al intervalo [0, longitud_cromosoma]
p_{cross}	probabilidad de reproducción de un individuo
p_{mut}	probabilidad de mutación de un individuo, en este caso se fija en el 1%.
t_i	tiempo aterrizaje del avión *i*

Inicio
    $g = 0$
    Generar aleatoriamente una población P_g

    Mientras ($g < g_{max}$)
        Desviaciones (P_g)
        Restricciones (P_g)
        Funciones objetivo (P_g)
        Ordenación NoDominada (P_g)
        Para cada ($F_i \in P_g$) hacer
            Diversidad (F_i)
        Fin Para
        $j = 0$
        Mientras ($|H| \leq np$)
            Seleccion (P_g)→ Pr_1 , Pr_2
            Crossover (Pr_1 , Pr_2) → (H_1, H_2)
            Mutación (H_1, H_2) →(H_1, H_2)
            $H_1, H_2 \rightarrow P_{g+1}$
            $j = j + 1$
        Fin mientras
        $g = g + 1$
    Fin Mientras

Codificación de cromosomas

La representación fenotípica de un cromosoma es:

Cromosoma 1	Avion1	Avion2	Avion3	...	AvionA
	t_1^1	t_2^1	t_3^1		t_A^1
Cromosoma 2	Avion1	Avion2	Avion3	...	AvionA
	t_1^2	t_2^2	t_3^2		t_A^2
…	…	…	…	…	…
Cromosoma N	Avion1	Avion2	Avion3	...	AvionA
	t_1^N	t_2^N	t_3^N		t_A^N

Tabla 6.1. Representación fenotípica de los cromosomas

Los cromosomas están formados por alelos, en este caso cada alelo representa un avión y el tiempo de aterrizaje estimado. El tiempo de aterrizaje de cada avión viene dado en minutos, es decir, se convertirán las horas en minutos, empezando desde las 00:00 horas que serán 0 minutos, y acumulativamente, por ejemplo, las 01:30 equivaldrán a 90minutos.

Los tiempos de aterrizaje de cada avión se codifican de forma binaria, es decir, representar cada cromosoma como uan cadena binaria. A cada 1 o 0 se le llama gen. Por ejemplo, el decimal 4 se refiere a la cadena binaria 100, si se necesita codificar dos variables que tienen valores comprendidos entre [0,15], entonces un cromosoma debe contener suficientes genes para la codificación, en este caso se necesitarían 8 genes ya que 15 representa la cadena binaria 1111 y al tener dos variables el cromosoma quedaría 1111 1111.

Póngase el caso que se tiene 3 aviones y para cada avión se debe definir su tiempo de inicio de aterrizaje. Si se codifica el tiempo entre 0 y 1440 minutos (1440 minutos = 24 horas) para cada avión se debería reservar 11 genes. En el caso de los 3 aviones, se necesitarían 33 genes por cromosoma, en el caso de 10 aviones se necesitarían 110 genes etc. Con esta codificación el algoritmo sería menos eficiente, porque necesitaría explorar muchas combinaciones de 0s y 1s. Por eso es importante encontrar una manera eficiente para codificar los tiempos de inicio de aterrizaje. En los algoritmos NSGA y NSGA-II que se detallan a continuación se utiliza el siguiente método:

1) Primero se define una ventana temporal de 15 minutos. Por ejemplo, a las 12:00 se buscan los aviones que deberían llegar entre las 12:00 y las 12:15.
2) Segundo, se codifican tiempos delta $\delta_{cromosoma_{avion_x}}$para cada avión. Por ejemplo, tienen que aparecer 3 aviones, avión A1 a las 12:01, avión A2 a las 12:03 y avión A2 a las 12:03. El algoritmo busca los tiempos de aterrizaje de los 3 aviones: A1= 12:01+δ_{A1}, A2=12:03+δ_{A2} y A3=12:03+δ_{A1}. Es decir, en lugar de codificar los tiempos de aterrizaje, se codifican los valores δ que varían para cada avión, por ejemplo, entre el intervalo [0,30] minutos. En este caso, únicamente se debería reservar 5 genes para cada

avión. A modo de explicación, siguiendo con el ejemplo de los 3 aviones, un cromosoma podría ser el siguiente: 00001 01001 00011, que se refiere a: A1 empieza a aterrizar a 12:01+1=12:02, A2 empieza a aterrizar a 12:03+9=12:12 y A3 a 12:03+3=12:06. Una vez calculados los tiempos de aterrizaje para cada avión de cada cromosoma, se deberá comprobar el cumplimiento o incumplimiento de las restricciones especificadas en el apartado "Planteamiento del problema" y en el caso de no cumplirlas se penalizará al cromosoma.

Ordenación basada en no-dominancia

Asignar $i = 1 \quad TP_g = P_g$
Mientras ($TP_g \neq \emptyset$)
    $F_i = i$
    Identificar soluciones no-dominadas en TP_g y asignarles F_i
    Asignar $TP_g = TP_g - F_i$
    $i = i + 1$
Fin Mientras
Para (cada $x \in P_g$)
    Asignar f_x= valor dummy | $f_x \in F_1 > f_y \in F_2 > \cdots > f_z \in F_n$
Fin Para

Diversidad

Para (cada $x, y \in F$)
    Calcular distancia d_{xy} :

$$d_{xy} = \sqrt{\sum_{x,y \in F_i} \sum_{k \in K} (z_k^{(x)} - z_k^{(y)})^2}$$

    Calcular el *sharing* de cada frente calculando el valor de la función *sharing* :

$$sh(d_{xy}) = \begin{cases} 1 - \left(\frac{d_{xy}}{\sigma_{share}}\right)^2, & si\ d_{xy} < \sigma_{share} \\ 0, & caso\ contrario \end{cases}$$

    Calcular el contador del nicho basándose en las distancias anteriormente calculadas.

$$nc_x = \sum_{y \in F_i} sh(d_{xy})$$

    Reajustar el *fitness* de cada solución *x*:

$$f'_x = \frac{f_x}{nc_x}$$

Fin Para

Selección por Torneo

$Pr_1 = \emptyset$
$Pr_2 = \emptyset$

$Pr\ \_op =$ Elegir aleatoriamente n individuos (P_g)
Ordenar individuos según f_n ($Pr\ \_op$)
$Pr_1 =$ mejor solución de $Pr\ \_op$
$Pr\ \_op =$ Elegir aleatoriamente n individuos (P_g)
Ordenar individuos según f_n ($Pr\ \_op$)
$Pr_2 =$ mejor solución de $Pr\ \_op$
Devolver (Pr_1 y Pr_2)

Crossover o reproducción

La reproducción o crossover se realiza utilizando el método de cruce en un solo punto. Aleatoriamente se elige un punto en el cual se intercambiaran los alelos entre los dos progenitores. La reproducción o crossover se aplica de acuerdo a la probabilidad de reproducción P_{cross}.

Punto de cruce nr

Progenitor1	A1	A3	A2	A4	A6	A5	A7
	t_1^1	t_2^1	t_3^1	t_4^1	t_5^1	t_6^1	t_7^1

Progenitor2	A1	A2	A3	A4	A7	A5	A6
	t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2	t_7^2

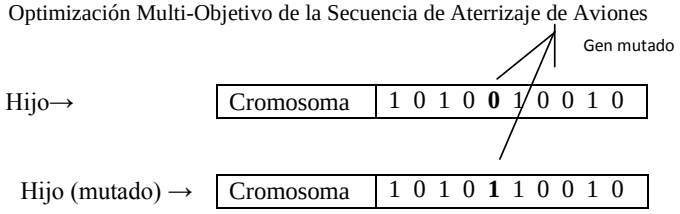
Hijo1	A1	A3	A2	A4	A7	A5	A6
	t_1^1	t_2^1	t_3^1	t_4^2	t_5^2	t_6^2	t_7^2

Hijo2	A1	A2	A3	A4	A6	A5	A7
	t_1^2	t_2^2	t_3^2	t_4^2	t_5^2	t_6^2	t_7^2

Pr_1
Pr_2
Generar número aleatorio r
Si ($r < p_{cross}$) hacer
    Generar punto de cruce (nr)
    Intercambiar genes a partir de la posición nr de Pr_1 y Pr_2 → H_1, H_2
    Calcular(f_{H_1}, f_{H_2})
    Devolver (H_1, H_2)
Sino hacer
    Devolver (Pr_1, Pr_2)
Fin Si

Mutación

El operador de mutación se aplica de manera individual a cada hijo. Consiste en la alteración aleatoria de cada gen componente del cromosoma de acuerdo a la probabilidad de mutación p_{mut}. En este caso, la representación genotípica es binaria, por lo cual, si un gen tiene valor 0 éste se cambia a 1 y viceversa.



Para cada($x_i \in H_j$) hacer
Generar número aleatorio r
Si ($r < p_{mut}$) hacer
modificar t_i
Fin Si
Fin Para

6.1.2. Cálculo paso a paso del algoritmo NSGA

A continuación se muestra una iteración del algoritmo NSGA para una población de 6. Los datos son:

Llegadas

	Tiempo Aparición	**Tiempo Objetivo**	**Tipo avión**
a1	610	614	H
a2	613	615	M
a3	613	615	H
a4	614	616	L

Salidas

a1_s	610		H
a2_s	612		M

Tabla 6.2. Datos cálculos de ejemplo del NSGA y NSGA-II

Tamaño de la población = 6 cromosomas
Generación aleatoria de la población P_0

Genotipo:

	a1	a2	a3	a4
cr1	00000	01001	01100	10000
cr2	00110	01110	10011	11000
cr3	01101	00110	11101	01110
cr4	01101	00001	10111	11010
cr5	10010	10100	10001	00100
cr6	10111	01011	10101	11100

Tabla 6.3. Población aleatoria codificada

Fenotipo (descodificación de la población). Se descodifican los tiempos obtenidos anteriormente para después sumarlo al tiempo de aterrizaje objetivo para obtener el tiempo de aterrizaje de cada avión.

	a1	**a2**	**a3**	**a4**
cr1	0	9	12	16
cr2	6	14	19	24
cr3	13	6	29	14
cr4	13	1	23	26
cr5	18	20	25	4
cr6	23	11	21	28

Tabla 6.4. Población descodificada

Tiempo aterrizaje:

	a1	**a2**	**a3**	**a4**
cr1	610	622	625	630
cr2	616	627	632	638
cr3	623	619	642	628
cr4	623	614	636	640
cr5	628	633	638	618
cr6	633	624	634	642

Tabla 6.5. Tiempos de aterrizaje

Calcular las desviaciones temporales respecto al tiempo de aterrizaje objetivo de cada avión para cada cromosoma

	a1	**a2**	**a3**	**a4**
cr1	4	-7	-10	-14
cr2	-2	-12	-17	-22
cr3	-9	-4	-27	-12
cr4	-9	1	-21	-24
cr5	-14	-18	-23	-2
cr6	-19	-9	-19	-26

Tabla 6.6. Desviaciones temporales respecto al tiempo de aterrizaje objetivo

Restricciones

A continuación se comprueba si con los tiempos de aterrizaje calculado se cumplen las restricciones se separación y de ocupación de pista.

Para comprobar que la se cumple la restricción de separación, primero se ordenan los aviones de cada cromosoma por orden de aterrizaje, luego se calcula el tiempo al que deberían aterrizar los aviones consecutivos al primero, y por último se comprueba que el tiempo de aterrizaje sea igual o superior al tiempo teórico de aterrizaje de cada avión.

Restricción de separación de seguridad:

Ordenación del aterrizaje de los aviones:

cr1	a1	a2	a3	a4
cr2	a1	a2	a3	a4
cr3	a2	a1	a4	a3
cr4	a2	a1	a3	a4
cr5	a4	a1	a2	a3
cr6	a2	a1	a3	a4

Tabla 6.7. Ordenación aviones según tiempo de aterrizaje

Calcular el tiempo teórico al que tendría que aterrizar el siguiente avión con tal de cumplir con la restricción:

	2º	**3º**	**4º**
cr1	612,62	623	628,27
cr2	612,62	628,00	635,27
cr3	620	626,27	629
cr4	615,00	624,6	639,27
cr5	619	630,62	634
cr6	625	634,6	637,27

Tabla 6.8. Tiempo teórico de aterrizaje

Comprobación del cumplimiento o no de la restricción. La tabla siguiente muestra si entre aterrizajes se cumple la restricción, si el valor es 1 se cumple la restricción, en caso contrario, la restricción no se cumple.

	1º-2º	**2º-3º**	**3º-4º**
cr1	1	1	1
cr2	1	1	1
cr3	1	1	1
cr4	1	1	1
cr5	1	1	1
cr6	1	-1	1

Tabla 6.9. Cumplimiento / incumplimiento restricción separación

Restricción de ocupación de pista:

Ordenación de los aviones por uso de pista (se debe que tener en cuenta, que en este caso también hay dos aviones que quieren salir)

Cálculo

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
cr1	a1	as_1	as_2	a2	a3	a4
cr2	as_1	as_2	a1	a2	a3	a4
cr3	as_1	as_2	a2	a1	a4	a3
cr4	as_1	as_2	a2	a1	a3	a4
cr5	as_1	as_2	a4	a1	a2	a3
cr6	as_1	as_2	a2	a1	a3	a4

Tabla 6.10. Ordenación de los aviones según tiempo aterrizaje y despegue

ar el tiempo teórico de salida de pista de cada avión para que se cumpla de restricción de ocupación de pista:

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
cr1	610,92	610,63	612,72	622,83	625,00	630,83
cr2	610,63	612,72	616,92	627,83	632,00	642,83
cr3	610,63	612,72	619,83	625,00	628,75	633,92
cr4	610,63	612,72	614,83	623,00	636,92	640,83
cr5	610,63	612,72	618,75	628,00	633,83	638,92
cr6	610,63	612,72	624,83	633,00	634,92	642,83

Tabla 6.11. Tiempo teórico de salida de pista

	1º-2º	**2º-3º**	**3º-4º**	**4º-5º**	**5º-6º**
cr1	-1	1	1	1	1
cr2	1	1	1	1	1
cr3	1	1	-1	1	-1
cr4	1	1	-1	1	1
cr5	1	1	1	1	-1
cr6	1	1	-1	1	1

Tabla 6.12. Cumplimiento / incumplimiento restricción de ocupación de pista

Comprobación del cumplimiento o no de la restricción. La tabla siguiente muestra si entre aterrizajes se cumple la restricción, si el valor es 1 se cumple la restricción, en caso contrario, la restricción no se cumple.

Cálculo de las funciones objetivo

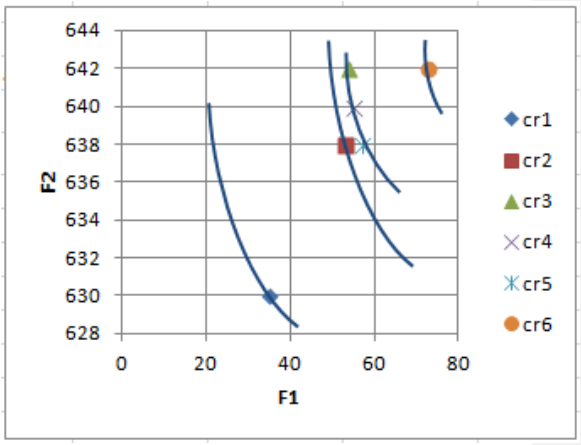
A continuación se calcula el valor de cada función objetivo para cada cromosoma y se le asigna el fitness según el número de cromosomas que domine parcialmente. Las funciones objetivo son:

F1 Minimización del coste total de la desviación temporal de los aterrizajes de los aviones

F2 Minimización del tiempo total de operación

	F1	F2	Fitness
cr1	35	630	4
cr2	53	638	3
cr3	54	642	2
cr4	55	640	2
cr5	57	638	2
cr6	73	642	1

Tabla 6.13. Valores F1 , F2 y fitness



Gráfica 6.1. Representación de F1, F2 y los diferentes frentes

La gráfica anterior representa los diferentes valores de cada cromosoma y los diferentes frentes. Una vez asignado el fitness a cada solución, se le aplica una penalización en el caso que se incumpla alguna restricción, por cada restricción que se incumpla se le resta una unidad al valor de fitness.

	Penalización restricciones	Fitness
cr1	1	3
cr2	0	3
cr3	3	1
cr4	1	1
cr5	1	1
cr6	2	1

Tabla 6.14. Penalizaciones de las restricciones

Diversidad

Para cada frente se debe reajustar el fitness de cada solución según la distancia entre soluciones y el parámetro share, cuyo valor se debe fijar.

$\sigma_{share} = 0{,}7$

Frente 1:

Únicamente hay dos soluciones, por lo tanto, el fitness de las soluciones no se modifica.

Frente 2:

	d_{xy}	sh_x
cr3-cr4	0,52	0,45
cr3-cr5	1,03	0,00
cr3-cr6	1,00	0,00
cr4-cr3	0,52	0,45
cr4-cr5	0,51	0,47
cr4-cr6	0,99	0,00
cr5-cr3	1,03	0,00
cr5-cr4	0,51	0,47
cr5-cr6	1,26	0,00
cr6-cr3	1,00	0,00
cr6-cr4	0,99	0,00
cr6-cr5	1,26	0,00

Tabla 6.15. Distancia crowding y sharing

Ajuste del fitness. Para cada cromosoma se ajusta el fitness en función del valor del contador de nicho, que es la suma del sharing calculada anteriormente por cromosoma.

	nC	**shared fitness**
cr1	-	3
cr2	-	3
cr3	0,45	2,23141264
cr4	0,92	1,08762835
cr5	0,47	2,12185782
cr6	-	1

Tabla 6.16. Shared fitness

Selección, reproducción y mutación

Para cada progenitor se eligen dos cromosomas aleatorios de la población y de entre éstos se elige aquel que tenga un mayor fitness. Para la reproducción de dos progenitores se genera un número aleatorio dentro del intervalo [0,1], si este valor es inferior a probabilidad de reproducción del cromosoma ($P_{cross} = 0{,}8$) se realiza la reproducción obteniendo dos hijos , en caso contrario, los progenitores pasaran a formar parte dela población de hijos. En el caso de la mutación, se genera un número aleatorio y si es inferior a la probabilidad de mutación ($P_{mut} = 0{,}01$) se modificará dicho gen. Este proceso se debe llevar a cabo hasta que el número de hijos sea igual al tamaño de la población.

Selección
Progenitor1
cr1
cr3
Progenitor 2
cr2

cr4

Reproducción

Punto de reproducción = 3

$p = 0{,}56$ → aplicar reproducción

cr1	00000	01001	01100	10000
cr2	00110	01110	10011	11100

hijo_1	00010	01110	10011	11100
hijo_2	00100	01001	01100	10000

Selección

Progenitor 3

cr4

cr2

Progenitor 4

cr3

cr5

Reproducción

Punto de reproducción = 16

$p = 0.87$ → no aplicar reproducción

cr2 = **hijo_3**	00110	01110	10011	11100
cr3= **hijo_4**	01111	00110	10100	01110

$Hijo\_3_1$ → $P_{mut} = 0{,}007$ (aplicar mutación al gen 1 del hijo_3)

cr2 = **hijo_3**	**1**0110	01110	10011	11100

Selección

Progenitor 5

cr1

cr3

Progenitor 6

cr5

cr4

Reproduccion

Punto de reproducción = 11

$p = 0.2$ → aplicar reproducción

cr1	00000	01001	01100	10000
cr5	10010	10100	10001	00100

hijo_5	00000	01001	00001	00100
hijo_6	10010	10100	11100	10000

Nueva población

Se sustituye la población inicial por la población de hijos obtenida, y ésta pasa a ser la población inicial de la siguiente iteración.

	a1	**a2**	**a3**	**a4**
cr1_1	00000	01110	10011	11100
cr1_2	00110	01001	01100	10000
cr1_3	10110	01110	10011	11100
cr1_4	01111	00110	10100	01100
cr1_5	00000	01001	00001	00100
cr1_6	10010	10100	11100	10000

Tabla 6.17.Población final

6.2. NSGA-II

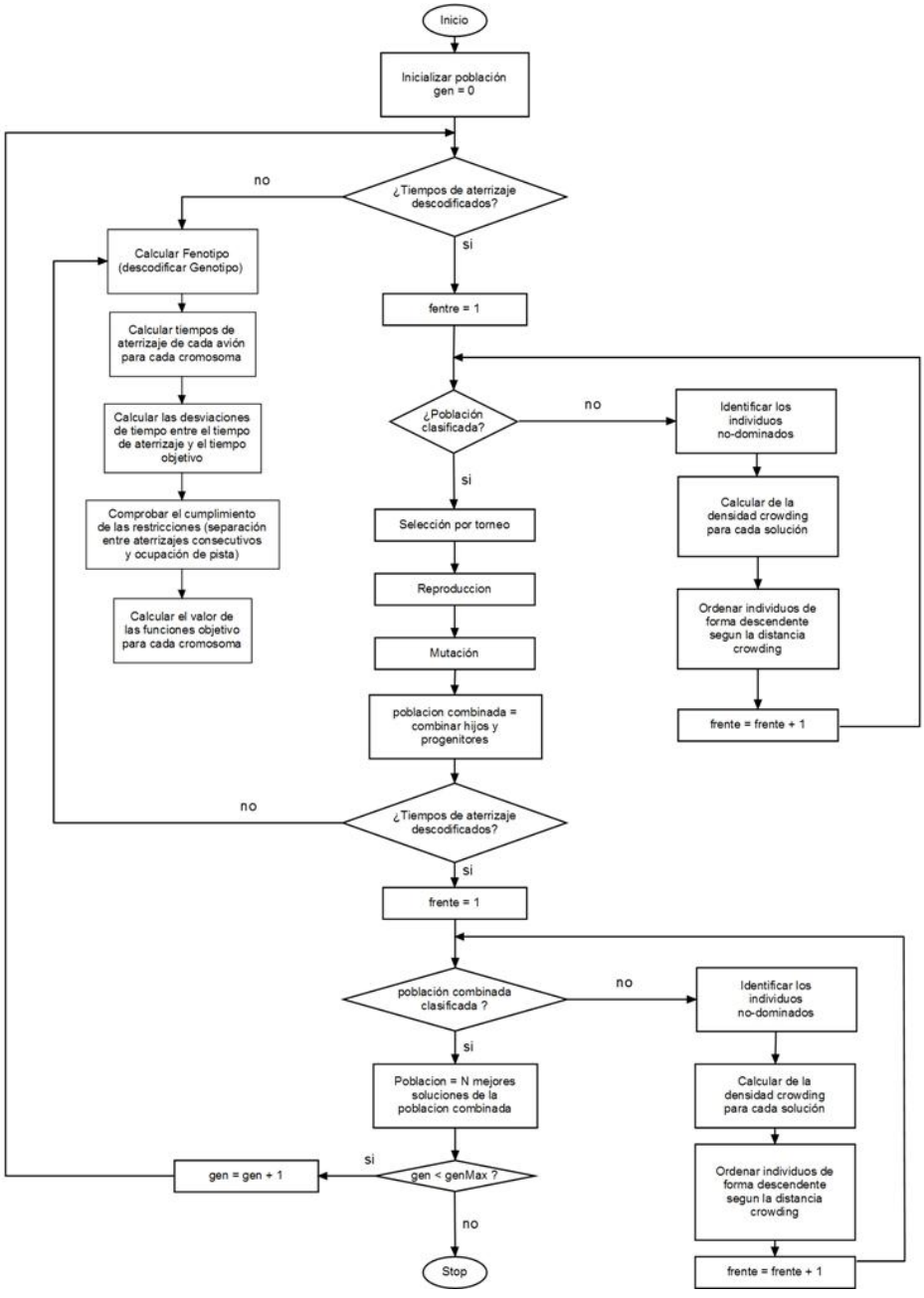


Ilustración 6.2. Diagrama de flujo NSGA-II

6.2.1. Pseudo-código NSGA-II

S conjunto de soluciones no-dominadas
l número de soluciones en el conjunto S

$\prec_n$ operador de comparacion de distancias crowding
M_i conjunto de soluciones dominadas por la solución i
ns_i numero de soluciones que dominan a la solución i
G frente de soluciones no-dominadas
$Rsep$ cumplimiento o incumplimiento de la restricción de distancia de seguridad entre dos aviones consecutivos que aterrizan. El valor 1 representa el cumplimiento de la restricción y el -1 el incumplimiento.
$Tsep$ suma del tiempo de aterrizaje y tiempo de separación para cada avión
$Rocup$ cumplimiento o incumplimiento de la restricción de ocupación de pista. El valor 1 representa el cumplimiento de la restricción y el -1 el incumplimiento.
$Tocup$ tiempo de ocupación de pista de cada avión.

Inicio
$g = 0$
Generar aleatoriamente una población P_0
Mientras ($g < g_{max}$)
$R_g = P_g \cup H_g$
Restricciones (R_g)
Funciones objetivo (R_g)
$F =$ Ordenación NoDominada (R_g)
$P_{g+1} = \emptyset$
$i = 0$
Mientras $|P_{g+1}| + |F_i| \leq N_p$ hacer
distancia crowding (F_i)
$P_{g+1} = P_{g+1} \cup F_i$
$i = i + 1$
Fin mientras
Ordenar P_{g+1} de forma descendiente mediante comparador crowding $\prec_n$
Seleccion (P_{g+1})→ Pr_1 , Pr_2
Crossover (Pr_1 , Pr_2) → (H_1, H_2)
Mutación (H_1, H_2) →(H_1, H_2)
$P_{g+1} = P_{g+1} \cup (H_1, H_2)$
//añadir restricciones, funciones objetivo, ordenacion no-dominada y distancia crowding
$P_{g+1} = P_{g+1} \cup F_i[1: (N_p - |P_{g+1}|]$
$g = g + 1$
Fin Mientras

Fin

Restricciones
$Rsep = \emptyset$
$Tsep = \emptyset$
$Rocup = \emptyset$
$Tocup = \emptyset$

Ordenar aviones por tiempo aterrizaje (R_g)
Para cada ($x \in R$)
Calcular ($Tsep$)
Calcular ($Tocup$)

Comprobar restricción de separación ($Rsep$, $Tsep$, R_g)

Comprobar restricción de separación ($Rocup, Tocup, R_g$)
Fin para

Funciones objetivo

$i = 1$
mientras $(i \leq n\_funciones\_objetivo)$
Para cada $(x \in R)$
Calcular z_x
Fin para
$i = i + 1$
fin mientras

Ordenación basada en no-dominancia

$F = \emptyset$
$i = 1$
Mientras $(R \neq 0)$ hacer
F_i =Soluciones no-dominadas (R)
$R = R - F_i$
$F = F \cup F_i$
$i = i + 1$
Fin mientras

Soluciones no-dominadas

para cada $(x \in R)$
para cada $(y \in R)$
si $(x \prec y)$ entonces
$M_x = M_x \cup \{y\}$
Sino si $(x \succ y)$ entonces
$ns_x = ns_x + 1$
Fin Si
Fin Para
Si $(ns_x = 0)$ entonces
$F_i = F_i \cup \{x\}$
Fin si
$i = 0$
mientras $(F_i \neq \emptyset)$
$G = \emptyset$
Para cada $(x \in F_i)$
Para cada $(y \in M_x)$
$ns_y = ns_y - 1$
Si $(ns_y = 0)$ entonces
$G = G \cup \{y\}$
Fin si
Fin para
Fin para
$i = i + 1$
$F_i = G$
Fin mientras

Crowding distance

$l = |S|$

Inicializar distancia
para cada($i \in 1 \dots l$) hacer
    inicializar distancia i
fin para
para cada ($j \in 1 \dots K$) hacer
    ordenar de forma ascendente según el valor de z_j
    asignar $S[1].distancia = \infty$ y $S[n_{si}].distancia = \infty$
    para cada ($j = 2$ hasta $j = n_{si} - 1$) hacer
    $S[i].distancia = S[i].distancia + (S[i+1].j - S[i-1].j)$
    Fin para
fin para

Comparador crowding

El operado de comparación $\prec_n$ guía en el proceso de selección en diferentes fases del algoritmo hacia la uniformidad de frente Pareto-óptimo. Se asume que cada individuo S_i tiene dos atributos: 1) ranking no-dominancia $S_{i_{rank}}$ y 2) la distancia *crowding* $S_{i_{distancia}}$. El operador de comparación se define como:

$s_i \prec_n s_j$ si ($S_{i_{rank}} < S_{j_{rank}}$) o ($\left(S_{i_{rank}} = S_{j_{rank}}\right)$ y ($S_{i_{distancia}} > S_{j_{distancia}}$))

6.2.2. Cálculos manuales del algoritmo NSGA-II

A continuación se muestra el funcionamiento de una iteración del algoritmo para una población pequeña. Los datos son los mismos que se han utilizado para la demostración del NSGA.

Tamaño de la población $P = 6$
Generación aleatoria de la población P_0
Genotipo:

	a1	**a2**	**a3**	**a4**
cr1	00000	01001	01100	10000
cr2	00110	01110	10011	11000
cr3	01110	00111	11101	01110
cr4	01101	00001	10111	11010
cr5	10010	10100	10001	00100
cr6	10111	01011	10101	11100

Tabla 6.18. Genotipo población NSGA-II

Fenotipo:

	a1	**a2**	**a3**	**a4**
cr1	0	9	12	16
cr2	6	14	19	24
cr3	14	7	29	14
cr4	13	1	23	26
cr5	18	20	25	4
cr6	23	11	21	28

Tabla 6.19. Fenotipo población NSGA-II

Tiempo de aterrizaje:

	a1	**a2**	**a3**	**a4**
cr1	610	622	625	630
cr2	616	627	632	638
cr3	624	620	642	628
cr4	623	614	636	640
cr5	628	633	638	618
cr6	633	624	634	642

Tabla 6.20. Tiempos de aterrizaje

Calculas las desviaciones temporales del tiempo de aterrizaje de cada avión para cada cromosoma:

	a1	**a2**	**a3**	**a4**
cr1	4	-7	-10	-14
cr2	-2	-12	-17	-22
cr3	-10	-5	-27	-12
cr4	-9	1	-21	-24
cr5	-14	-18	-23	-2
cr6	-19	-9	-19	-26

Tabla 6.21. Desviaciones de tiempo respecto al tiempo de aterrizaje objetivo

Restricciones

Al igual que el algoritmo anterior, NSGA, se comprueba si los cromosomas obtenidos anteriormente cumplen o no las restricciones impuestas. Primero se ordenan los aviones de cada cromosoma según el tiempo de aterrizaje.

cr1	a1	a2	a3	a4
cr2	a1	a2	a3	a4
cr3	a2	a1	a4	a3
cr4	a2	a1	a3	a4
cr5	a4	a1	a2	a3
cr6	a2	a1	a3	a4

Tabla 6.22. Ordenación aviones según tiempo de aterrizaje

Restricción de separación:

Se calculan los tiempos de aterrizaje más el tiempo de separación de seguridad:

	2º	**3º**	**4º**
cr1	612,62	623	628,27
cr2	618,62	628,00	635,27
cr3	621	627,27	629
cr4	615,00	624,6	639,27
cr5	619	630,62	634
cr6	625	634,6	637,27

Tabla 6.23. Tiempo teórico de aterrizaje

La tabla de a continuación muestra si se cumple la restricción de separación entre el aterrizaje de dos aviones consecutivos.

	1º-2º	**2º-3º**	**3º-4º**
cr1	1	1	1
cr2	1	1	1
cr3	1	1	1
cr4	1	1	1
cr5	1	1	1
cr6	1	-1	1

Tabla 6.24. Cumplimiento / incumplimiento restricción de separación

Restricción de ocupación de pista:

La pista únicamente puede estar ocupada por un avión. Por lo cual, hay que comprobar que no aterrice ningún avión mientras la pista este ocupada. Para ello, calculamos en que momento abandonará la pista el avión y se comprueba que el tiempo de aterrizaje del siguiente avión sea igual o superior al de ocupación de pista. En este caso, añadimos dos aviones que quieren salir, los datos son los siguientes:

Salidas	**Tiempo salida**	**Tipo avión**	**Tiempo pista (seg)**
a1_s	610	H	38
a2_s	612	M	43

Orden aviones

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
cr1	a1	a1_s	as_2	a2	a3	a4
cr2	as_1	as_2	a1	a2	a3	a4
cr3	as_1	as_2	a2	a1	a4	a3
cr4	as_1	as_2	a2	a1	a3	a4
cr5	as_1	as_2	a4	a1	a2	a3
cr6	as_1	as_2	a2	a1	a3	a4

Tabla 6.25. Ordenación de los aviones según tiempo aterrizaje y despegue

Tiempos salida de pista de cada avión:

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
cr1	610,92	610,63	612,72	622,83	625,92	630,75
cr2	610,63	612,72	616,92	627,83	632,92	642,75
cr3	610,63	612,72	620,83	624,92	628,75	633,92
cr4	610,63	612,72	614,83	623,92	636,92	640,75
cr5	610,63	612,72	618,75	628,92	633,83	638,92
cr6	610,63	612,72	624,83	633,92	634,92	642,75

Tabla 6.26. Tiempo teórico de salida de pista

Se comprueba si se cumple la restricción o no:

	1º-2º	2º-3º	3º-4º	4º-5º	5º-6º
cr1	-1	1	1	1	1
cr2	1	1	1	1	1
cr3	1	1	-1	1	-1
cr4	1	1	-1	1	1
cr5	1	1	1	1	-1
cr6	1	1	-1	1	1

Tabla 6.27. Cumplimiento / incumplimiento restricción separación de pista

Funciones objetivo

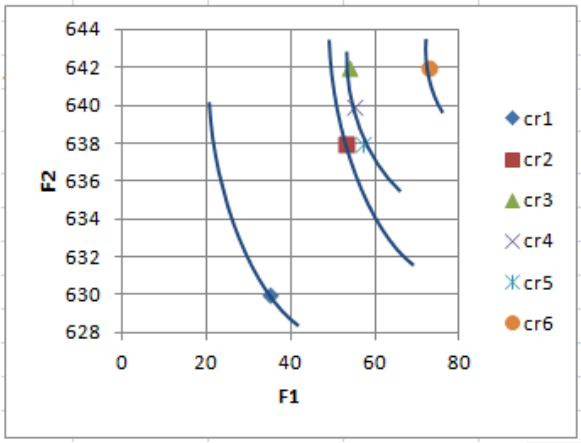
F1 minimización del coste total de la desviación de tiempo

F2 minimización del tiempo de aterrizaje del ultimo avión

Formatat: Tipus de lletra: (Per defecte) Times New Roman, Negreta

	F1	F2
cr1	35	630
cr2	53	638
cr3	54	642
cr4	55	640
cr5	57	638
cr6	73	642

Tabla 6.28. Valores F1 y F2



Gráfica 6.2. Representación F1, F2 y diferentes frentes

Soluciones no dominadas

Una vez calculados los valores de las funciones objetivos se identifican las soluciones no-dominadas y se identifican los diferentes frentes. El valor ns_x de cada solución se reajusta según el número de restricciones que cada cromosoma incumpla y se identifican los diferentes frentes.

	M_x	ns_x	Nº restricciones incumplidas	ns_x	Frente
cr1	cr2, cr3,cr4,cr5,cr6	0	1	1	1
cr2	cr4, cr2	1	0	1	1
cr3	cr4,cr6	1	0	1	1
cr4	cr6	3	0	3	2
cr5	cr4,cr6	1	0	1	1
cr6	-	5	1	6	3

Tabla 6.29. Relación entre soluciones

Para cada frente, si está formado por más de 2 cromosomas, se calcula la *distancia crowding* de las soluciones. Primero se normalizan los valores de las funciones objetivos y luego, se calcula la distancia.

	F1	**F2**
cr1	0,00	0,00
cr2	0,47	0,67
cr3	0,45	1,00
cr4	0,53	0,83
cr5	0,58	0,67
cr6	1,00	1,00

Tabla 6.30. Valores normalizados F1 y F2

Frente 1:

	dF1	**dF2**	**cd**
cr1	1	1	2
cr2	0,13	1,00	1,13
cr3	0,47	0,33	0,81
cr5	0,53	1,00	1,53

6.31. Distancia crowding

Frente 2: una única solución

Frente 3: una única solución

Selección, reproducción y mutación

Para la reproducción primero se seleccionan dos progenitores aleatoriamente y se escoge aquel que pertenezca al frente de menor valor, así se obtiene el primer progenitor, y se repite el mismo proceso para obtener el segundo progenitor. Una vez se obtiene la pareja, se genera un número aleatorio, si éste es menor a la probabilidad de reproducción, se reproduce la pareja y se obtienen dos hijos, en caso contrario, los progenitores se copian en la población de hijos. Para cada gen de cada hijo se genera un número aleatorio, si éste es inferior a la probabilidad de mutación del gen, éste se muta. El proceso de selección, reproducción y mutación se repite hasta que la población de hijos sea del mismo tamaño que la población inicial.

Selección
Progenitor 1
cr1
cr5
Progenitor 2
cr2
cr3
Reproducción
Punto de reproducción = 11

$p = 0{,}62$ → aplicar reproducción

cr1	00000	01001	01100	10000
cr2	00110	01110	10011	11000

hijo_1	00000	01001	00011	11000
hijo_2	10010	10100	11100	10000

Selección
Progenitor 3
cr5
cr2
Progenitor 4
cr2
cr3
Reproducción
Punto de reproducción = 16
$p = 0{,}3$ → aplicar reproducción

cr5	10010	10100	10001	00100
cr3	01110	00111	10011	11000

hijo_3	10010	10100	10001	01000
hijo_4	01110	00111	10100	10100

Selección
Progenitor 5
cr6
cr4
Progenitor 6
cr2
cr1
Reproducción
Punto de reproducción = 8
$p = 0{,}8$ → no aplicar reproducción

cr4 = hijo_5	01101	00001	10111	11010
cr1 = hijo_6	00000	01001	01100	10000

$Hijo\_6_3$ → $P_{mut} = 0{,}007$ (aplicar mutación al gen 3 del hijo_6)

hijo_6	00**1**00	01001	01100	10000

Fenotipo población Hijo

	a1	**a2**	**a3**	**a4**
hijo_1	0	9	3	24
hijo_2	18	20	28	16
hijo_3	18	20	17	8
hijo_4	14	7	20	20
hijo_5	13	1	23	26
hijo_6	4	9	12	16

Tabla 6.32. Fenotipo población Hijo

Tiempo aterrizaje

	a1	**a2**	**a3**	**a4**
hijo_1	610	622	616	618
hijo_2	628	633	641	630
hijo_3	628	633	630	622
hijo_4	624	620	633	634
hijo_5	623	614	636	640
hijo_6	614	622	625	630

Tabla 6.33. Tiempos de aterrizaje

Calculo de las desviaciones temporales del aterrizaje de los aviones de cada cromosoma respecto su tiempo de aterrizaje objetivo

	a1	**a2**	**a3**	**a4**
hijo_1	4	-7	-1	-2
hijo_2	-14	-18	-26	-14
hijo_3	-14	-18	-15	-6
hijo_4	-10	-5	-18	-18
hijo_5	-9	1	-21	-24
hijo_6	0	-7	-10	-14

Tabla 6.34. Desviaciones de tiempo respecto al tiempo de aterrizaje objetivo

Restricciones

Se comprueba si la nueva población de hijos obtenida cumple o no las restricciones. Primero se ordenan según el tiempo de aterrizaje los cromosomas y después se realiza la comprobación.

hijo_1	a1	a3	a4	a2
hijo_2	a1	a4	a2	a3
hijo_3	a4	a1	a3	a2
hijo_4	a2	a1	a3	a4
hijo_5	a2	a1	a3	a4
hijo_6	a1	a2	a3	a4

Tabla 6.35. Ordenación aviones según tiempo aterrizaje

Restricción de seguridad:

Se calcula el tiempo teórico mínimo el cual el siguiente avión debería realizar el aterrizaje de modo que la distancia de separación de seguridad se cumpla.

	2º	**3º**	**4º**
cr1	611,60	619,27	619,15
cr2	631,27	631,15	634,00
cr3	623,00	629,60	632,62
cr4	621,00	627,60	636,27
cr5	615,00	624,60	639,27
cr6	616,62	623,00	628,27

Tabla 6.36. Tiempo teórico de aterrizaje

La tabla siguiente, muestra si la restricción se cumple o no, si el valor es 1 implica que la restricción se cumple, en caso contrario, el valor que aparece es el -1.

	1º-2º	**2º-3º**	**3º-4º**
hijo_1	1	-1	1
hijo_2	-1	1	1
hijo_3	1	1	1
hijo_4	1	1	-1
hijo_5	1	1	1
hijo_6	1	1	1

Tabla 6.37. Cumplimiento / incumplimiento restricción de separación

Restricción de ocupación de pista:

Primero se calcula el tiempo en que un avión abandona la pista, y se comprueba que la pista no esté ocupada por un avión en el momento en el que aterriza. En este caso, se debe tener en cuenta que tenemos dos aviones que tienen salida prevista para un determinado tiempo, por lo tanto también se debe comprobar que mientras uno de estos dos aviones ocupe pista para realizar el despegue, no haya ningún avión que aterrice hasta que el avión haya despegado.

La ordenación de uso de pista teniendo en cuenta los aviones que despegan es la siguiente:

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
hijo_1	as_1	a1	as_2	a3	a4	a2
hijo_2	as_1	as_2	a1	a4	a2	a3
hijo_3	as_1	as_2	a4	a1	a2	a3
hijo_4	as_1	as_2	a2	a1	a4	a3
hijo_5	as_1	as_2	a2	a1	a3	a4
hijo_6	as_1	as_2	a2	a3	a1	a4

Tabla 6.38.Ordenación aviones según tiempo de aterrizaje y despegue

Cálculo del tiempo en el que cada avión abandona la pista:

	1º	**2º**	**3º**	**4º**	**5º**	**6º**
hijo_1	610,63	610,92	612,72	616,92	618,75	622,83
hijo_2	610,63	612,72	628,92	630,75	633,83	641,92
hijo_3	610,63	612,72	622,75	628,92	630,92	633,83
hijo_4	610,63	612,72	620,83	624,92	633,92	634,75
hijo_5	610,63	612,72	614,83	623,92	636,92	640,75
hijo_6	610,63	612,72	614,92	622,83	625,92	630,75

Tabla 6.39. Tiempo teórico salida de pista

La tabla siguiente, muestra si la restricción se cumple o no, si el valor es 1 implica que la restricción se cumple, en caso contrario, el valor que aparece es el -1.

	1º-2º	**2º-3º**	**3º-4º**	**4º-5º**	**5º-6º**
hijo_1	-1	1	1	1	1
hijo_2	1	1	1	1	1
hijo_3	1	1	1	1	1
hijo_4	1	1	1	1	1
hijo_5	1	1	1	1	1
hijo_6	1	1	1	1	1

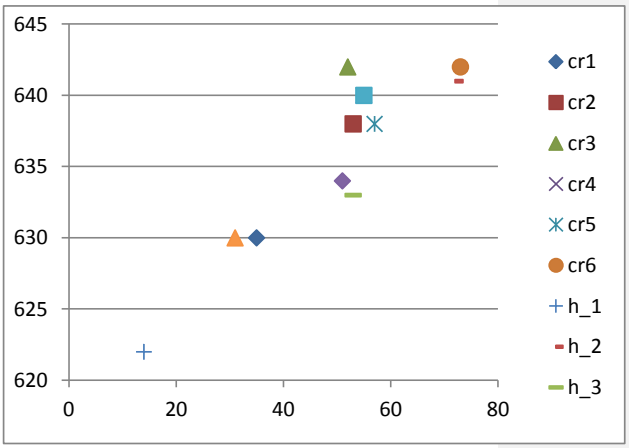
Tabla 6.40. Cumplimiento / incumplimiento restricción ocupación de pista

El NSGA-II a diferencia del NSGA utiliza un mecanismo de elitismo. Una vez comprobado el cumplimiento de las restricciones se combinan ambas poblaciones pertenecientes a la generación actual. A partir de este momento se utiliza la población combinada. Se calcula el valor de las funciones objetivos y se identifican las soluciones no-dominadas y los frentes pertenecientes a dicha población para finalmente seleccionar aquellos individuos que formaran la población inicial de la siguiente generación. La selección se realizara en base al valor del frente en el que se encuentren las soluciones (partiendo de que en el frente 1 se encuentran las mejores soluciones y los siguientes contendrán peores soluciones).

Funciones objetivo de la población combinada

	F1	F2
cr1	35	630
cr2	53	638
cr3	54	642
cr4	55	640
cr5	57	638
cr6	73	642
h_1	14	622
h_2	72	641
h_3	53	633
h_4	51	634
h_5	55	640
h_6	31	630

Tabla 6.41. Valores F1 y F2



Gráfica 6.3. Representación F1 y F2

Soluciones no dominadas

Se identifican las soluciones no dominadas y los frentes de la población combinada.

	M_x	ns_x	Nº restricciones incumplidas	ns_x	Frente
cr1	cr2, cr3, cr4, cr5, cr6, h_2, h_3, h_4, h_5,	1	1	2	2
cr2	cr4, cr6, h_2, h_5	4	0	4	3
cr3	cr6	4	0	4	3
cr4	cr6, h_2	6	0	6	5
cr5	cr6	4	0	4	3
cr6	-	11	1	12	6
h_1	cr1,cr2,cr3,cr4,cr5,cr6, h_2, h_3, h_4, h_5, h_6	0	2	2	2
h_2	cr6	7	1	8	
h_3	cr4, cr5, cr6, h_2, h_5	3	1	4	3
h_4	cr2, cr3, cr4, cr5, cr6, h_2	3	1	4	3
h_5	cr6	5	0	5	4
h_6	cr2, cr3, cr4, cr5, cr6, h_2, h_3, h_4, h_5	1	0	1	1

Tabla 6.42. Relación entre soluciones

Para cada frente se calcula la *distancia crowding* de las soluciones que lo componen. Para calcular dicha distancia primero deben normalizarse los valores de las funciones objetivo.

	F1	F2
cr1	0,36	0,40
cr2	0,66	0,80
cr3	0,64	1,00
cr4	0,69	0,90
cr5	0,73	0,80
cr6	1,00	1,00
h_1	0,00	0,00
h_2	0,98	0,95
h_3	0,66	0,55
h_4	0,63	0,60
h_5	0,69	0,90
h_6	0,29	0,40

Tabla 6.43. Valores normalizados F1 y F2

Frente 1: una única solución

Frente 2: una única solución

Frente 3:

	dF1	dF2	cd
cr2	0,02	0,40	0,42
cr3	0,03	0,20	0,23
cr5	0,34	0,40	0,74
h_3	0,07	0,60	0,67
h_4	0,64	0,25	0,89

Tabla 6.44. Distancias *crowding*

Frente 4: una única solución

Frente 5: una única solución

Frente 6: una única solución

Al ordenar la población combinada según los frentes y la distancia crowding calculada la población queda de la siguiente manera:

	ns	cd
h_6	1	0
h_1	2	0
cr1	2	0
h_4	4	0,89
cr5	4	0,74
h_3	4	0,67
cr2	4	0,42
cr3	4	0,23
h_5	5	0
cr4	6	0
h_2	8	0
cr6	12	0

Tabla 6.45. Población progenitores e hijos ordenada según los frentes y distancia crowding

Al inicio del algoritmo, se ha fijado el tamaño n de la población, por lo tanto, de la población combinada ordenada, se seleccionarán los mejores n cromosomas que formarán la población inicial para la siguiente iteración.

Población inicial de la siguiente iteración

cr1_1 (h_6)	00100	01001	01100	10000
cr2_1 (h_1)	00000	01001	00011	11000
cr3_1 (cr1)	00000	01001	01100	10000
cr4_1 (h_4)	01110	00111	10100	10100
cr5_1 (cr5)	10010	10100	10001	00100
cr6_1 (h_3)	10010	10100	10001	01000

Tabla 6.46. Población inicial siguiente generación

6.3. Conclusiones

Mediante el paso-a-paso de los dos algoritmos se puede apreciar las diferencias que existen entre el NSGA y el NSGA-II, siendo el segundo una versión mejorada. El hecho de que el NSGA-II utilice un método elitista permite comparar los cromosomas de población de progenitores y de la población de hijos, para después seleccionar los mejores, permitiendo conservar soluciones que eran potencialmente buenas. A diferencia del NSGA-II, el NSGA no aplica ningún método elitista y únicamente sustituye la población de progenitores por la población de hijos sin tener en cuenta si hay soluciones en la población de progenitores, haciendo que así se pierdan soluciones potencialmente buenas. Otra diferencia a destacar es el en NSGA hay que definir el parámetro σ_{share} a criterio del usuario, dependiendo del valor que tome la solución variará.

7. Experimentos numéricos y resultados

En esta sección se presentan los experimentos de análisis de los dos algoritmos llevados a cabo. Primero se presenta diseño del experimento y luego con los datos obtenidos se realiza un test ANOVA para comparar diferentes aspectos de los dos algoritmos propuestos con la herramienta Minitab. El output completo tanto del diseño del experimento como de los análisis estadísticos se encuentran en los anexos.

7.1. Diseño del experimento

Para ayudar a la hora de determinar cuál de los dos algoritmos es mejor para resolver el problema descrito, se realiza un diseño experimental. El primer paso es determinar los factores a analizar y sus niveles:

	Niveles	
Factores		
Tipo de algoritmo	NSGA	NSGA-II
Numero de generaciones	100	200
Tamaño de la población	50	100
Crossover rate	0,5	0,8

Tabla 7.1. Factores para el diseño de experimento (DOE)

A la hora de realizar el diseño del experimento se especifica que sea un diseño factorial, de dos niveles y con replica. En este caso, el diseño cuenta con 4 factores con dos niveles cada uno, por lo tanto, hay que realizar un total de 160 pruebas en el orden establecido en *RunOrder* y con la configuración de factores establecida. La tabla 7.2. muestra parte del resultado del diseño de experimento a realizar.

StdOrder	RunOrder	CenterPt	Blocks	Algorithm	Num iteratione	Population size	Crossover rate
68	1	1	1	NSGA-II	200	50	0,5
112	2	1	1	NSGA-II	200	100	0,8
158	3	1	1	NSGA-II	100	100	0,8
130	4	1	1	NSGA-II	100	50	0,5
117	5	1	1	NSGA	100	100	0,5
75	6	1	1	NSGA	200	50	0,8
56	7	1	1	NSGA-II	200	100	0,5
49	8	1	1	NSGA	100	50	0,5
35	9	1	1	NSGA	200	50	0,5
128	10	1	1	NSGA-II	200	100	0,8
141	11	1	1	NSGA	100	100	0,8
72	12	1	1	NSGA-II	200	100	0,5
107	13	1	1	NSGA	200	50	0,8
109	14	1	1	NSGA	100	100	0,8

Tabla 7.2. Muestra del experimento a realizar

Los conjuntos de datos con los que se van a realizar los análisis de resultados cumplen las especificaciones de los métodos estadísticos, que son:

(i) Independencia de las observaciones
(ii) Residuos siguen una distribución normal
(iii) Homogeneidad de las varianza

7.2. Caso 1: análisis del tiempo de ejecución de los algoritmos NSGA y NSGA-II

Una de las medidas de calidad a la hora de comparar dos algoritmos es el tiempo que tardan en obtener los resultados y la relevancia a que tienen los factores para las soluciones de los dos algoritmos. La Tabla 7.3. muestra los resultados de la prueba t de Student, con estos resultados se puede determinar si los algoritmos tienen el mismo tiempo de ejecución o no. Las hipótesis nula y alternativa son las siguientes:

H_0: La diferencia entre la media de tiempos de ejecución esntre NSGA y NSGA-II es 0.

H_A: La diferencia entra la media de tiempos de ejecución entre NSGA y NSGA-II es diferente a 0 de forma significante.

$\alpha = 0.05$

Paired T-Test and CI: Tiempo NSGA; Tiempo NSGA-II

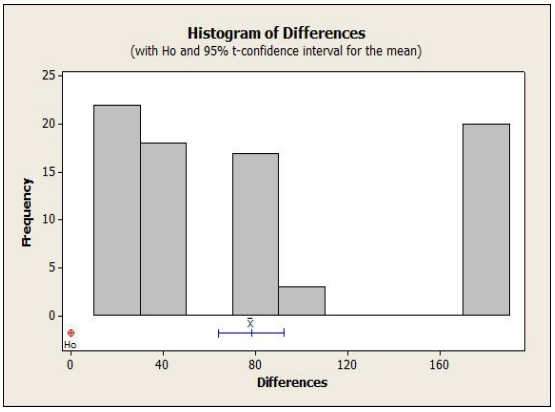
Paired T for Tiempo NSGA - Tiempo NSGA-II

	N	Mean	StDev	SE Mean
Tiempo NSGA	80	98,78	76,48	8,55
Tiempo NSGA-II	80	20,58	12,80	1,43
Difference	80	78,20	63,80	7,13

95% CI for mean difference: (64,00; 92,39)
T-Test of mean difference = 0 (vs not = 0): T-Value = 10,96 P-Value = 0,000

Tabla 7.3. Resultados T-test para la comparación de tiempo de ejecución de los algoritmos

Formatat: Inglés (Estados Unidos)
Formatat: Tipus de lletra: 12 pt, (Asiàtica) Japonés (Japón), (Altres) Inglés (Estados Unidos)
Formatat: Inglés (Estados Unidos)

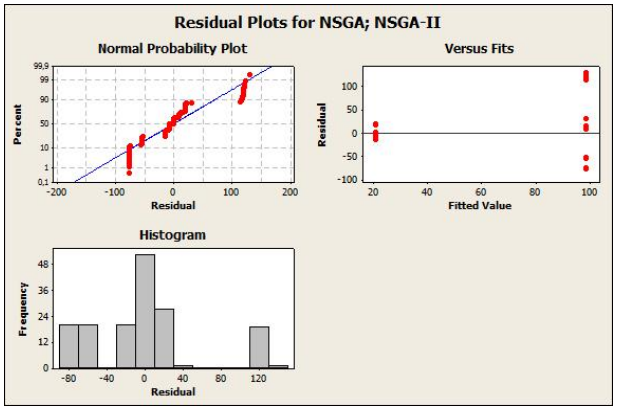


Gráfica 7.1. Histograma de las diferencias entre los tiempos de ejecución de los algoritmos NSGA y NSGA-II

One-way ANOVA: NSGA; NSGA-II

```
Source   DF      SS      MS      F      P
Factor    1  244589  244589  81,36  0,000
Error   158  474993    3006
Total   159  719582

S = 54,83   R-Sq = 33,99%   R-Sq(adj) = 33,57%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level     N   Mean  StDev  -------+---------+---------+---------+--
NSGA     80  98,78  76,48                            (---*---)
NSGA-II  80  20,58  12,80  (---*---)
                          -------+---------+---------+---------+--
                                30        60        90       120

Pooled StDev = 54,83


Grouping Information Using Tukey Method

          N   Mean  Grouping
NSGA     80  98,78  A
NSGA-II  80  20,58    B

Means that do not share a letter are significantly different.


Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons

Individual confidence level = 95,00%


NSGA subtracted from:

          Lower  Center   Upper  --+---------+---------+---------+-------
NSGA-II  -95,32  -78,20  -61,07  (-----*-----)
                                 --+---------+---------+---------+-------
                                 -90       -60       -30         0
```

Table 7.4.. One-Way ANOVA tiempo ejecución algoritmos



Gráfica 7.2. Residuos ANOVA tiempos de ejecución de los algoritmos

Como se muestra en la tabla 7.3. se rechaza la hipótesis nula ya que el $p - valor = 0{,}00 < \alpha = 0.05$, por lo que se concluye que los algoritmos son significativamente diferentes en cuanto a los tiempos de ejecución. En la tabla 7.4.. mediante el método Tukey se comprueba que los dos algoritmos son diferentes ya que las medias no pertenecen al mismo grupo. También podemos concluir, que el NSGA-II tiene un tiempo medio de ejecución de 20,58 segundos frente a 98,78 segundos del NSGA, es decir, el algoritmo NSGA-II es más rápido que el NSGA. La gráfica 7.1. es el histograma de las diferencias entre tiempos de ejecución de los algoritmos, la forma que debería seguir es la de campana . En la gráfica 7.2. se representan varias gráficas para los residuos, la gráfica *Normality Probabilty* demuestra que los residuos no siguen una distribución normal, esto puede deberse a la naturaleza de los datos, del histograma de la misma gráfica, se deduce que los datos asimétricos estadísticamente, aunque dibuja una campana, lo que indica que los residuos siguen una distribución normal.

7.3. Caso 2: análisis del mejor conjunto de soluciones encontradas por los algoritmos NSGA y NSGA-II:

La Tabla 7.5. muestra los resultados de la prueba t de Student para comparar la calidad de los frentes finales que proporciona cada algoritmo. Los p-valores de ambos test son inferiores a 0,05 por lo que podemos concluir que existen diferencias significativas entre los frentes finales de cada algoritmo. Las gráficas 7.3. y 7.4. muestran la semejanza de las distribuciones a una distribución normal. La gráfica 7.3. muestra un desplazamiento hacia el lado positivo de la gráfica (por encima de la media), lo que indica una alta asimétrica positiva, por lo que se produce una mayor concentración de datos en eje derecho de la gráfica. La gráfica 7.4. no presenta ningún tipo de desplazamiento respecto a la media, y se puede intuir la forma de campana de los resultados.

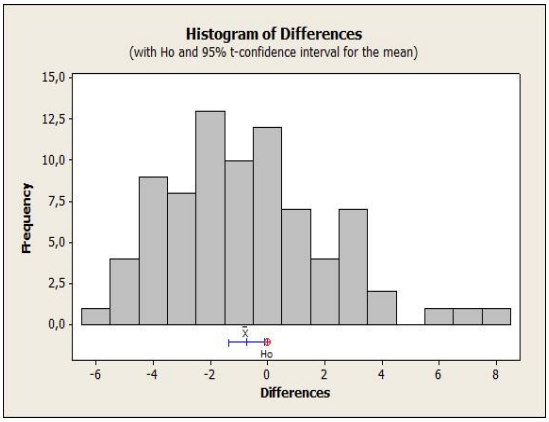
Paired T-Test and CI: F1 NSGA; F1 NSGA-II

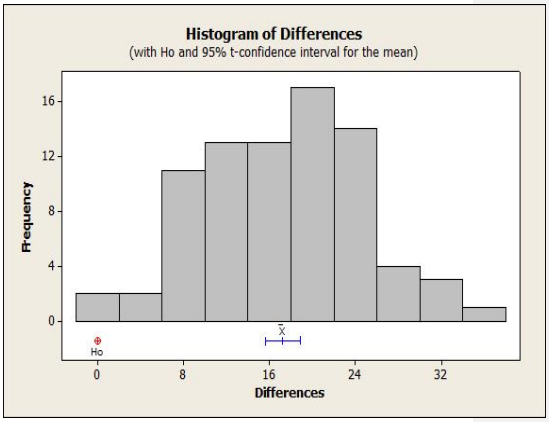
```
Paired T for F1 NSGA - F1 NSGA-II

              N     Mean  StDev  SE Mean
F1 NSGA      80  847,075  3,352    0,375
F1 NSGA-II   80  847,812  2,497    0,279
Difference   80   -0,738  2,833    0,317


95% CI for mean difference: (-1,368; -0,107)
T-Test of mean difference = 0 (vs not = 0): T-Value = -2,33  P-Value = 0,022
```

Paired T-Test and CI: F2 NSGA; F2 NSGAII

```
Paired T for F2 NSGA - F2 NSGAII

              N    Mean  StDev  SE Mean
F2 NSGA      80  59,614  6,610    0,739
F2 NSGAII    80  42,369  4,234    0,473
Difference   80  17,245  7,436    0,831


95% CI for mean difference: (15,590; 18,900)
T-Test of mean difference = 0 (vs not = 0): T-Value = 20,74  P-Value = 0,000
```

Tabla 7.5.Resultados T-test para la comparación de la calidad de los frentes





Gráfica 7.3. y 7.4. Histogramas de diferencias de los algoritmos para F1 y F2 respectivamente

7.3.1. One-Way ANOVA

Otro aspecto determinante referente a los algoritmos presentados es el efecto de los factores en la solución. Para saber si los factores tienen afectan al resultado, se ha realizado un test One-Way ANOVA para cada algoritmo, los resultados están resumidos en las tablas 7.6. y 7.7. para el NSGA y 7.8. y 7.9. para el NSGA-II. Las hipótesis nula y las alternativas son las siguientes:

$H_{0_{iteracion}}$: el parámetro número de iteraciones no influye en el resultado

$H_{0_{poblacion}}$: el parámetro tamaño de la población no influye en el resultado

$H_{0_{reproduccion}}$: el factor probabilidad de reproducción no influye en el resultado

	F1					F2				
	DF	SS	MS	F	P	DF	SS	MS	F	P
Parámetros										
Número iteraciones	1	0,2	0,2	0,02	0,894	1	75,1	75,1	1,74	0,192
Tamaño población	1	84,8	84,8	8,25	0,005	1	102,8	102,8	2,40	0,126
Probabilidad de reproducción	1	3,1	3,1	0,27	0,604	1	6,4	6,4	0,14	0,705

Tabla 7.6. Análisis de los parámetros principales del NSGA

	F1			F2		
Niveles de los parámetros	LS Mean	Grupos Homogéneos		LS Mean	Grupos homogéneos	
Numero iteraciones						
IT: 100	847,12	X		60,583	x	
IT: 200	847,02	X		58,645	x	
Tamaño población						
P: 50	848,10	X		60,747	x	
P: 100	846,04		x	58,480	x	
Probabilidad reproducción						
Prep: 0,5	847,27	X		58,896	x	
Prep: 0,8	846,88	X		59,331	x	

Tabla 7.7. Test comparación de rangos para los parámetros del algoritmo NSGA

Las tablas 7.6. y 7.7. son un resumen del análisis ANOVA llevado a cabo para determinar la influencia de los parámetros sobre los resultado del NSGA. De ellas cabe destacas que el único parámetro que influye en el resultado del NSGA es, según los niveles definidos, el tamaño de la población, cuyo p-valor es 0,006, por lo tanto se rechaza la $H_{0_{poblacion}}$. Estas conclusiones se corroboran en la tabla 7.7. en la que se muestra la media para cada nivel de cada factor, y si existe o no homogeneidad de grupo entre niveles. Al analizar los diferentes niveles de cada factor se puede determinar la influencia que tienen sobre los resultados.

	F1					F2				
	DF	SS	MS	F	P	DF	SS	MS	F	P
Parámetros										
Número iteraciones	1	0,03	0,03	0	0,948	1	50,6	50,6	2,89	0,093
Tamaño población	1	45,22	45,22	7,89	0,006	1	438,6	438,6	35,01	0
Probabilidad de reproducción	1	1,37	1,37	0,22	0,642	1	19,5	19,5	1,09	0,3

Tabla 7.8.. Análisis de los parámetros principales NSGA-II

	F1			F2		
Niveles de los parámetros	LS Mean	Grupos Homogéneos		LS Mean	Grupos homogéneos	
Numero iteraciones						
IT: 100	847,83	X		43,164	x	
IT: 200	847,79	X		41,574	x	
Tamaño población						
P: 50	848,56	X		60,747	x	
P: 100	846,06		x	58,480	x	
Probabilidad reproducción						
Prep: 0,5	847,68	X		59,896	x	
Prep: 0,8	847,94	X		59,331	x	

Tabla 7.9. Test comparación de rangos para los parámetros del algoritmo NSGA-II

Las tablas 7.8. y 7.9. son un resumen del análisis ANOVA llevado a cabo para determinar la influencia de los parámetros sobre los resultado del NSGA-II. De la tabla 7.8. se observa que el parámetro tamaño de la población es el único que influye en el resultado del NSGA, según los niveles de los parámetros definidos. El tamaño de la población es el parámetro más relevante debido al valor del ratio F = 35,01, en comparación al ratio del número de iteraciones que es F = 4,05 (referente a F2).

7.3.2. MANOVA

Para comprobar los resultados obtenidos se ha realiza un test MANOVA con el que también se comprobará si entre los factores existe interacción. Los resultados del MANOVA se resumen en las tablas 7.10. y 7.11. para el NSGA y el NSGA-II, respectivamente. Las hipótesis nula y las alternativas son las siguientes:

H_0: el factor no influye en la solución

H_A: el factor influye en la solución

	F1					F2				
	DF	SS	MS	F	P	DF	SS	MS	F	P
Parámetros										
n iteraciones	1	0,2	0,2	0,02	0,89	1	75,14	75,14	1,74	0,192
tamaño población	1	84,84	84,84	7,98	0,006	1	102,83	102,83	2,38	0,127
crossover	1	3,07	3,07	0,29	0,593	1	6,38	6,38	0,15	0,701
Interacciones										
n iteraciones*tamaño poblacion	1	11,81	11,81	1,11	0,295	1	1,98	1,98	0,5	0,831
n iteraciones*crossover	1	1,78	1,78	0,17	0,684	1	107,74	107,74	2,49	0,119
tamaño población*crossover	1	9,7	9,7	0,91	0,343	1	1,19	1,19	0,03	0,869
error	73	776,03	10,63			73	3156,56	43,24		
total	79	887,44				79	3451,86			

Tabla 7.10. Análisis de los parámetros principales del NSGA

La tablas 7.10. es un resumen del análisis MANOVA llevado a cabo para determinar la influencia de los parámetros sobre los resultado del NSGA. Las hipótesis nula y alternativa son:

$H_{0_{iteracion}}$: el parámetro número de iteraciones no influye en el resultado

$H_{0_{poblacion}}$: el parámetro tamaño de la población no influye en el resultado

$H_{0_{reproduccion}}$: el factor probabilidad de reproducción no influye en el resultado

De la tabla 7.10. cabe destacar que el único parámetro que influye en el resultado del NSGA es, según los niveles definidos, el tamaño de la población, cuyo p-valor es 0,006, por lo tanto se rechaza la $H_{0_{poblacion}}$. Este resultado coincide con los test llevado a cabo en el apartado anterior. En la tabla 7.7. se muestra la media para cada nivel de cada factor, y si existe o no homogeneidad de grupo entre niveles. Al analizar los diferentes niveles de cada factor se puede determinar la influencia que tienen sobre los resultados. Otro hecho importante es que no existe interacción entre ninguno de los parámetros, el p-valor de los factores es superior a 0,05, por lo tanto, aceptamos la hipótesis nula de que existe no interacción entre los factores.

	F1					F2				
	DF	SS	MS	F	P	DF	SS	MS	F	P
Parámetros										
n iteraciones	1	0,027	0,027	0	0,946	1	50,57	50,57	4,5	0,037
tamaño población	1	45,215	45,215	7,97	0,06	1	438,61	438,61	39,01	0
crossover	1	1,37	1,37	0,24	0,625	1	19,45	19,45	1,73	0,192
Interacciones										
n iteraciones & tamaño población	1	19,169	19,169	3,38	0,07	1	0,23	0,23	0,02	0,885
n iteraciones & crossover	1	11,369	11,369	2	0,161	1	2,48	2,48	0,22	0,64
tamaño población & crossover	1	0,862	0,862	0,15	0,698	1	83,84	83,84	7,46	0,008
error	73	414,398	10,63			73	820,72	11,24		
total	79	492,409				79	1415,91			

Tabla 7.11. Análisis de los parámetros principales NSGA-II

La tablas 7.11. es un resumen del análisis ANOVA llevado a cabo para determinar la influencia de los parámetros sobre los resultado del NSGA-II. Las hipótesis nula y alternativa son:

$H_{0_{iteracion}}$: el parámetro número de iteraciones no influye en el resultado

$H_{0_{poblacion}}$: el parámetro tamaño de la población no influye en el resultado

$H_{0_{reproduccion}}$: el factor probabilidad de reproducción no influye en el resultado

De la tabla 7.11. destaca que los parámetros número de iteraciones y tamaño de la población influyen en el resultado del NSGA, según los niveles de los parámetros definidos. El tamaño de la población es el parámetro más relevante debido al valor del ratio F = 39,01, en comparación al ratio del número de iteraciones que es F = 4,05 (referente a F2). En la tabla7. 9. podemos observar como para F2 los parámetros de número de iteraciones y tamaño de población pertenecen al mismo grupo, esto es debido a que los test que se han realizado para comparar los rangos de cada parámetro se realizan individualmente, a diferencia del MANOVA que se ha realizado con todos los parámetro. En la tabla 7.11. se observa que únicamente existe interacción entre los parámetros de tamaño de población y probabilidad de reproducción, el p-valor de los factores es inferior a 0,05, por lo tanto, rechazamos la hipótesis nula de que no existe interacción entre los factores.

Comentari [H1]: Should I mention it? I'm not sure about that statement. But the tables 7.2.4 and 7.2.6. are made with the one-way ANOVA for each objective function and I've read that this can cause an accumulation of errors. Is that right?

7.4. Conclusiones

Los experimentos realizados para obtener los resultados fueron implementados en un procesador Intel(R) Core(TM) i7-3610Q CPU 2.30GHz RAM 8.00Gb. Con dichos resultados se ha llevado a cabo un análisis estadístico de los diferentes parámetros involucrados en los algoritmos evolutivos NSGA y NSGA-II. Existe una diferencia significativa entre las medias de tiempo de ejecución de los algoritmos, siendo en NSGA-II más rápido, el NSGA tiene un tiempo de ejecución medio de 98,78 segundos, frente a una media de 20,58 segundos del NSGA-II. Al analizar la influencia de los diferentes parámetros a las soluciones, se concluye que para los niveles de los parámetros definidos, sobre las soluciones que proporciona el NSGA afecta el tamaño de la población, y sobre las soluciones del NSGA-II afecta la el número de iteraciones y el tamaño de la población.

8. Resultados y Conclusiones

La limitación latente de los aeropuertos de absorber la creciente demanda de vuelos hace que se investiguen métodos para aprovechar de la manera más eficiente los recursos disponibles de éste. Uno de los factores más relevantes en cuanto a eficiencia es el uso de pista del aeropuerto y la coordinación de aterrizajes y despegues de aviones de forma segura. El problema secuencia de aterrizajes y despegues se conoce como *Aircraft Landing Scheduling Problem*, e intenta encontrar secuencias que satisfagan las necesidades de tráfico y cumpla con las restricciones de seguridad.

En este proyecto se ha formulado el problema de secuencia de aterrizaje de aviones como un problema multi-objetivo que tiene por funciones objetivo minimizar la desviación total entre el tiempo de aterrizaje y el tiempo objetivo de los aviones y minimizar el tiempo de aterrizaje del utlimo avión. Se ha realizado un nuevo intento de resolverlo, utilizando algoritmos MOEAs (*Multi-Objetive Evolutionary Algorithm*). Para resolver el problema se ha revisado la literatura existente sobre los diferentes métodos de resolución así como los diferentes métodos de optimización de problemas multi-objetivo.

De todos los métodos analizados para la resolución del problema multi-objetivo se han seleccionado los algoritmos NSGA y NSGA-II. El NSGA es capaz de encontrar un conjunto se soluciones y una mejor convergencia mejores de la solución aproximada del verdadero frente Pareto-optimo comparado con los algoritmos PAES y SPEA. El NSGA también se ha seleccionado como primera versión del NSGA y así poder analizar las mejoras y diferencias entre ambas versiones.

El paso-a-paso de los dos algoritmos permite ver las diferencias que existen entre el NSGA y el NSGA-II, siendo el segundo una versión mejorada del primero. La gran diferencia que existe entre un algoritmo y otro es que el NSGA-II aplica un método elitista que se aplica en el NSGA-II, ya que una vez obtenida la población de hijos, ésta se une con la población de progenitores para así seleccionar los mejores cromosomas evitando así la pérdida de soluciones potencialmente buenas. Otro factor importante a destacar es el hecho de que en el NSGA se tiene que definir el valor del parámetro σ_{share} ya que dependiendo del valor que se le otorgue se obtendrán un conjunto de soluciones u otro.

Finalmente se han realizado análisis estadísticos de los resultados de los algoritmos. Los experimentos realizados para obtener los resultados, tal como se ha referenciado, fueron implementados en un procesador Intel(R) Core(TM) i7-3610Q CPU 2.30GHz RAM 8.00Gb. Con dichos resultados se ha llevado a cabo un análisis estadístico de los diferentes parámetros involucrados en los algoritmos evolutivos NSGA y NSGA-II. Existe una diferencia significativa

entre las medias de tiempo de ejecución de los algoritmos, siendo en NSGA-II más rápido, el NSGA tiene un tiempo de ejecución medio de 98,78 segundos, frente a una media de 20,58 segundos del NSGA-II. Al analizar la influencia de los diferentes parámetros a las soluciones, se concluye que para los niveles de los parámetros definidos, sobre las soluciones que proporciona el NSGA afecta el tamaño de la población, y sobre las soluciones del NSGA-II afecta la el número de iteraciones y el tamaño de la población.

Bibliografía

[1] JE Beasley, M Krishnamoorthy, YM sharaiha, D Abramson: Displacement Problem and dynamically scheduling aircraft landings // Journal of the Operational Research Society. - 2004 - 55 p. 54 – 64.

[2] G. Bencheikh, J. Boukachour, A. El Hilali Alaoui, F. El Khoukhi: Hybrid method for aircraft landing scheduling based on a Job Shop formulation // International Journal of Computer Science and Network Security.-2009.- VOL. 9.- No. 8.- p.78 – 88

[3] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva, G. Mills: Computing Optimal Schedules for Landing Aircraft // 12th National Conference of the Australian Society for Operations Research.-1995.

[4] Vic Ciesielski, Paul Scerri: An anytime Algorithm of Aircraft Landing Times Using Genetic Algorithms// RMIT University, Department of Computer Science, August 1998

[5] Ke Tang, Zai Wang, Xiabin Cao, Juan Zhang : A Mutli-Objective Approach to Aircraft Landing Scheduling Problems // IEEE Congress on Evolutionary Computation 2008

[6] Zhao-Tong Huang, Xue-Yan Song, Ji-Zhou Sun, Zhi-Zeng Li: Aircraft Landing Scheduling Based on Semantic Agent Negotiation Mechanism // School of computer Science and Technology, Tianjin University. ICIC 2012, LNCS 7389. p 483 – 489

[7] Hamsa Balakrishnan, Bala Chandran: Scheduling Aircraft Landing under Constrained Position Shifting // Ph.D. Thesis. American Institute of Aeronautics and Astronautics. August 2006

[8] Amir Salehipour, Mohammad Modarres, Leila Moslemi Naeni: An efficient hybrid meta-heuristic for aircraft landing problem // Computers and Operations Research.-2013.Vol.40.-No.1.-p. 207-213

[9] Napalkova L. Development and application of multi-objective optimization methods. PhD Thesis. Riga Technical University. 2010.

[10] Fonseca C.M., Fleming P.J. Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalisation// Forrest S. (Ed.): Proceedings of the Fifth International Conference on Genetic Algorithms.- Morgan Kauffman Publishers,1993.-p. 416-423.

[11]Srinivas N., Deb K. Multi-objective function optimisation using non-dominated sorting genetic algorithms// Evolutionary computation journal.-1994.- Vol.2.-No.3.- p. 221-248.

[12] Deb K., Agrawal S., Pratab A., Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II// Schoenauer M., Deb K., Rudolph G., Yao X., Lutton E., Merelo J.J., Schwefel H.P.(eds.): Proceedings of the Parallel Problem Solving from Nature VI Conference. - Paris, France: Springer, 2000.-p. 849–858.

[13]Horn J., Nafploitis N., Goldberg D. A niched Pareto genetic algorithm for multi-objective optimization// Proceedings of the First IEEE Conference on Evolutionary Computation.-1994.- p. 82-87.

[14]Zitzler E., Thiele L. An evolutionary algorithm for multi-objective optimisation: The strength Pareto approach// Technical report No.43, Zürich, Switzerland: Computer

Formatat: Inglés (Estados Unidos)

Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 1998.

[15]Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm// Technical Report No.103.-Swiss Federal Institute of Technology, 2001.

[16]Knowles J.D., Corne D.W. Approximating the non-dominated front using the Pareto archived evolution strategy// Evolutionary Computation Journal.-2000.- Vol.8.- No.2.-p. 149-172.