Inteligencia Artificial

Informe Final: Problema Aircraft Landing Scheduling Problem

Javiera Paz Villarroel Toloza

26 de diciembre de 2021

Evaluación

Mejoras 2da Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El *Aircraft Landing Scheduling Problem* busca crear una planificación al encontrar la secuencia de aterrizaje de los aviones en sus respectivas pistas de aterrizaje, que haga el mejor uso de estas minimizando los costos, pues cada avión tiene un tiempo ideal de aterrizaje y desviarse de este implica un costo. La secuencia está sujeta a restricciones, tales como cumplir con la ventana de tiempo de aterrizaje de cada avión y la separación de tiempo entre aviones. Se presentan planteamientos del problema en la historia y modelos matemáticos para los enfoques estático y dinámico, junto a la implementación de Backtracking con CBJ, experimentación con distintas instancias y sus resultados. En las conclusiones, se señala la importancia de enfrentar este problema y las ventajas y desventajas de la implementación, promoviendo la investigación en métodos estocásticos o heurísticas de selección.

1. Introducción

El *Aircraft Landing Scheduling Problem* es un problema que consiste en planificar los aterrizajes de los aviones, decidiendo el tiempo de aterrizaje para cada avión de forma que cada uno aterrice dentro de una ventana de tiempo, cumpliendo con restricciones sobre el tiempo de separación entre los aterrizajes de los aviones, con el objetivo de minimizar el costo total por la desviación del tiempo ideal de aterrizaje de cada avión, tiempo que se asocia a la velocidad donde el avión gasta menos combustible.

En las últimas decadas, el tráfico aéreo ha experimentado un gran aumento y un crecimiento sostenido, por ejemplo, en el 2018, los servicios aéreos regulares transportaron un total de 4.300 millones de pasajeros, presentando un aumento del 6,1 % con respecto al año anterior, o en 2017 donde hubo un aumento del 7,9 % [1], mientras que para la carga aérea hubo un aumento del 7,7 %. Por otro lado, uno de los aeropuertos más ocupado, el Atlanta-Hartsfield-Jackson, manejó 104 millones de pasajeros en 2017 [6]. Estas cifras demuestran cómo el transporte aéreo tiene una

gran influencia en el mundo, transportando pasajeros de todo el mundo y permitiendo las importaciones, sin embargo, mientras la demanda aumenta o se mantiene, el tráfico aéreo presenta a una limitación pues las pistas de aterrizaje son limitadas y se convierten en el principal cuello de botella de las operaciones aéreas, al mismo tiempo que los aterrizajes de cada avión deben cumplir con restricciones mencionadas anteriormente, significando una dificultad en la planificación de todos los aviones, pues una mala decisión puede conllevar costos altos en combustible debido a las desviaciones que pueda tener el avión con respecto a su tiempo ideal de aterrizaje, y no tan solo eso, pues el retraso de un vuelo puede provocar retrasos de otros debido a que no podrán aterrizar en la misma pista. Por ello, el problema abordado es una importante área en las operaciones de tráfico aéreo, pues permitirá la creación de planificaciones que obtengan una mejor rentabilidad, que sean factibles y confiables, brindando un menor tiempo de espera para los pasajeros, un mejor flujo del aeropuerto y menores costos.

El propósito de este estudio es presentar el *Aircraft Landing Scheduling Problem*, comenzando por definir el problema y comprender sus implicancias en la vida real, para luego exponer los distintos enfoques que se ha presentando en la literatura hasta hoy, con respecto a la disponibilidad de los datos, incertidumbre, número de pistas de aterrizaje, función objetivo y sus restricciones, como también se analizan sus métodos de resolución asociados, ya sean métodos exactos o estocásticos. Dentro de los enfoques de los modelos presentados, en la sección de modelado se profundiza en dos de ellos, donde se presenta su modelo matemático: Modelo estático y dinámico del *Aircraft Landing Scheduling Problem*. Además, se presenta una implementación para encontrar solución a este problema, donde se utiliza la técnica completa *Backtracking con Conflict-based backjumping*, explicando la representación de soluciones y descripción del algoritmo utilizado. Se realizan experimentos utilizando esta implementación haciendo uso de distintas instancias, para luego analizar y comparar los resultados obtenidos. Finalmente, se concluye sobre lo expuesto en este estudio, presentando nuevas ideas para continuar con las investigaciones del problema.

2. Definición del Problema

En los aeropuertos se controla el tráfico aéreo utilizando radares, por lo que cada avión necesita tener asignado un *tiempo de aterrizaje* [8], es decir, el momento en que el avión aterrizará y además, necesita ser asignado a una pista de aterrizaje. Este tiempo de aterrizaje debe estar dentro de una ventana de tiempo específica, limitada por un *tiempo más temprano* y *tiempo más tardío* [8], el primero representando el tiempo más temprano si el avión vuela a su máxima velocidad, mientras que el segundo representa el tiempo más tardío en que puede aterrizar a su velocidad más eficiente en combustible, sin embargo, cada avión tiene una velocidad preferida y económica que se conoce como velocidad de crucero, el tiempo asociado a esta velocidad y su aterrizaje se denomina *tiempo objetivo* [8]. Si el control aéreo requiere que el avión ralentice o aumente su velocidad, existe un *costo asociado*, el cual aumenta mientras la diferencia entre el tiempo objetivo y tiempo de aterrizaje aumente, existiendo un costo en caso de que aterrice antes del tiempo objetivo y otro en caso de que aterrice después del tiempo objetivo.

Por otro lado, existen los *tiempos de separación* entre aterrizaje de un avión y otro, delimitados por consideraciones aerodinámicas; por ejemplo, un Boeing 747 genera mucha turbulencia de aire, y un avión volando muy cerca podría perder su estabilidad aerodinámica, por ello, el Boeing 747 necesita un delay de tiempo antes de que los otros aviones puedan aterrizar [8].

Estos tiempos de separación son uno de los factores que determinan la capacidad de la pista de aterrizaje, y los estándares más comúnes de esta separación consisten en:

- La *separación radar*, una separación de distancia mínima que debe cumplirse entre aviones bajo un área de control de radar. La Organización de Aviación Civil Internacional (ICAO) específica la separación vertical para cada avión volando bajo las reglas de vuelo instrumental como 300 m bajo un nivel de vuelo 290 (altitud 29000 ft) y 600 m en o por

encima de este nivel. Si dos aviones están separados por una separación vertical menor a la mínima, se requiere una separación longitudinal, la cual son 5 millas nauticas, o 3 millas nauticas en areas congestionadas. [11]

- La *separación de estela-vórtice*, que asegura que ningún avión sea afectado por la turbulencia de los vórtices generados por los aviones de adelante, como se menciono anteriormente, y especialmente durante despegues y aterrizajes. Esta separación entre dos aviones consecutivos depende de la categoría de turbulencia del avión que va delante y el que va detrás. [11]

El problema *Aircraft Landing Scheduling Problem* ocurre principalmente en los aeropuertos más congestionados, donde se necesita hacer un uso óptimo de las pistas de aterrizaje, manejando de forma efectiva y segura los aterrizajes de un flujo continuo de aviones entrando al rango del radar hacia las pistas asignadas. Si bien las funciones objetivo varían respecto al modelo utilizado para el problema, esta suele ser **minimizar el costo total asociado a la desviación de cada avión de su tiempo objetivo**, cumpliendo con restricciones que fueron mencionadas anteriormente tales como: la ventana de tiempo específica, los tiempos de separación y el aterrizaje de cada avión. Por otro lado, los parámetros asociados a este modelo son: el tiempo más temprano, tardío y tiempo objetivo de cada avión, el costo de cada avión al aterrizar antes o despúes del tiempo objetivo y los tiempos de separación entre cada par de aviones, y además, el número de pistas de aterrizaje disponibles y el número de aviones totales.

Dentro del interés de optimizar el uso de pistas de aterrizaje, existen distintas variaciones del problema:

- *Aircraft Landing Problem* (ALP), que considera solo los aterrizajes [11]
- *Aircraft Take-off Problem* (ATP), que considera solo los despegues [11]
- *Aircraft Scheduling Problem* (ASP), que considera la secuencia y planificación de despegues y aterrizajes [11]

Los dos primeros consisten en primero asignar una pista disponible para cada avión listo para aterrizar (o despegar para ATP), y luego asignando una hora programada de aterrizaje (o despegue) para cada uno.

Dentro de las variaciones del problema mencionadas anteriormente, el ASP es un problema más realista, pues los controladores aéreos deben tratar simultáneamente con el tráfico de despegues y aterrizajes, sin embargo, este problema es conocido por ser **NP-difícil** [10], y por lo tanto, el tiempo de solución utilizando métodos exactos no escala bien con el tamaño del problema, en este caso, el número de aviones, por lo que representa un desafío para los controladores aéreos ya que deben tratar con un número creciente de aeronaves, siendo una de las principales dificultades que enfrenta este problema.

Por otro lado, al aumentar las pistas de aterrizaje, se agrega la variable de determinar la pista de aterrizaje apropiada a la que deben aterrizar los aviones, cumpliendo aún con tiempos de separación, dependiendo de si dos pares de aviones aterrizan en la misma pista o no, agregando más complejidad al problema. Además, al agregar más pistas de aterrizaje, crece también el espacio de búsqueda, provocando una explosión combinatorial tal como ocurre al aumentar el número de aeronaves.

Otras dificultades que enfrenta son situaciones que pueden ocurrir en la práctica, tales como las condiciones climáticas que pueden afectar los vuelos, u otras situaciones que provoquen un retraso de los vuelos e incluso, su cancelación, por lo tanto, afectan también la planificación de la secuencia de aterrizaje que se tenía preparada en un inicio.

3. Estado del Arte

Existen algunas técnicas básicas utilizadas por controladores para la secuencia de aviones:

1. *First-Come First-Served* (FCFS) es una técnica de secuenciación que planifica los aviones basados en su tiempo estimado de llegada (ETA) y su tiempo estimado de despegue (ETT). Para un avión llegando, el secuenciador FCFS calcula los horarios programados de aterrizaje (SLT) basados en el orden dado por los ETA, obtenido cuando un avión entra al radar del aeropuerto, y teniendo en consideración las restricciones impuestas de separación. Esta técnica es ampliamente utilizada en la práctica, pues es fácil de implementar, sin embargo, en aeropuertos congestionados, es poco probable que provea secuencias óptimas. [11]

2. *Constrained Position Shifting* (CPS) es un concepto introducido en 1976 por *Dear*, y permite a los aviones desviarse del FCFS hasta un número máximo de cambios de posición. Por ejemplo, si un avión ocupa la quinta posición en la secuencia de aterrizaje, y el número máximo de cambios de posición permitido es dos, entonces el avión puede ser re-programado en las positiones 3, 4, 5, 6 o 7. [11]

El primer acercamiento a resolver este problema ocurrió en *1976* por *Dear* [11], con la técnica que se mencionó anteriormente, y a partir de ese momento, distintos modelos y soluciones han salido a la luz.

Existen distintas formulaciones matemáticas propuestas en la literatura, las cuales pueden ser clasificadas acorde a:

- **Disponibilidad de los datos de entrada**: Los modelos son *estáticos* cuando los inputs se conocen con antelación, mientras que es *dinámico* cuando algunos inputs son desconocidos por el horizonte de tiempo considerado. [11]

- **Incertidumbre en los valores de parámetro**: Los modelos pueden ser *deterministas* o *bajo incertidumbre*. [11]

- **Número de pistas de aterrizaje**: Existe los modelos que consideran el caso de una pista de aterrizaje, mientras que otros consideran *múltiples pistas*. En este caso, existen varias configuraciones posibles, como pistas paralelas cuyas líneas centrales son paralelas. [11]

- **Función objetivo**: Variaciones como maximizar el throughput de las pistas de aterrizaje (número de operaciones por hora), minimizar el máximo delay, minimizar las desviaciones ponderadas de los tiempos objetivo. [11]

- **Restricciones**: Las restricciones fundamentalmente consideradas se relacionen a la segura separación de los tiempos, pero también se agregan las restricciones de ventanas de tiempo, o de precedencia. [11]

Los modelos más citado dentro del ALSP consisten en formulaciones *Mixed-Integer Programming* (MIP). Uno de ellos es la formulación **mixed-integer zero-one con un enfoque estático** propuesta por *Beasley* [8], pudiendo ser aplicado tanto para modelar con solo despegues o aterrizajes (modo segregado), o con los dos (modo mixto). Uno de los beneficios de estos modelos es que permiten realizar comparaciones cuantitativas explícitas entre políticas alternativas de operación de pistas dentro de un marco de optimización [8]. En este modelo, se asume que se minimiza el costo total, donde el costo para cada avión está relacionado linealmente a la desviación del tiempo objetivo. Este modelo matemático se presentará en la siguiente sección.

Otros modelos [9] han adoptado un enfoque utilizando el **job-shop scheduling** para resolver el ALSP, donde las pistas de aterrizajes representan las máquinas idénticas y los aviones

representan los trabajos [8]. El tiempo más temprano asociado a cada avión (trabajo) es el tiempo listo del trabajo. Se suele asumir que el tiempo más tardío es lo suficientemente largo como para no tener importancia. El tiempo de procesamiento de un trabajo en particular (avión) en una máquina (pista de aterrizaje) depende de:

- Solo el trabajo que le sigue en la misma máquina (separación sucesiva)
- Todos los otros trabajos que siguen en la misma máquina (separación completa)

En los **modelo dinámicos**, se considera que las decisiones deben ser constantemente revisadas debido a los cambios operacionales en el entorno, pues para tomar una nueva decisión, se debe considerar explícitamente la decisión previa, teniendo un problema de decisión genérico, el *displacement problem*, que surge cuando se tiene que tomar una secuencia de decisiones, y cada decisión consecutiva involucra un enlace a la decisión previa. Para ello, se utiliza una *displacement function* que cuantifica el efecto de desplazar cada variable de decisión desde su valor de solución previo y conocido hasta un nuevo valor desconocido [7]. La formulación de este modelo se presenta en la sección siguiente.

3.1. Soluciones

A continuación, se presenta un gráfico con el número de artículos más recientes, dependiendo del problema que aborden y re-agrupado en cuatro tipos de metodologías: programación dinámica, programación mixta-entera, métodos metaheurísticos/heurísticos y aprendizaje reforzado [11]

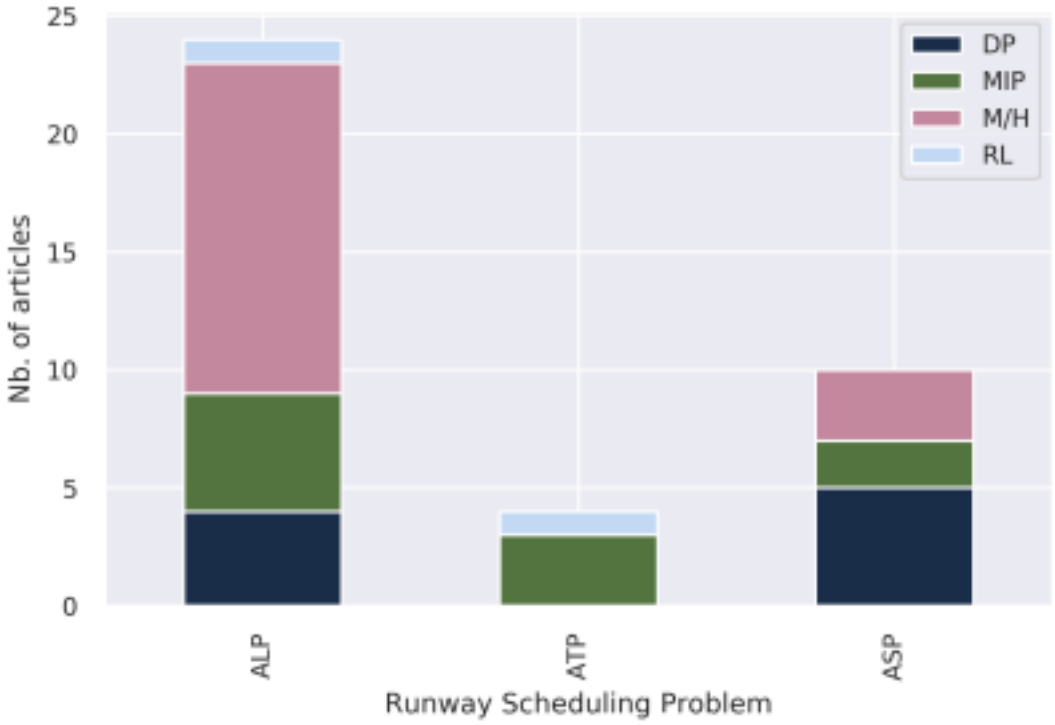

Figura 1: Número de artículos utilizando distintos métodos de solución como Dynamic Programming (DP), Mixed-Integer Programming (MIP), Metaheuristics/Heuristics (M/H), and Reinforcement Learning (RL) para las variantes del problema [11]

Se puede observar que una gran cantidad de trabajos han utilizado heurísticas y metaheurísticas. Una posible explicación de esta tendencia es la complejidad del problema, favoreciendo los métodos estocásticos capaces de proveer soluciones de buena calidad, a diferencia de métodos exactos que suelen requerir altos tiempos de cómputo. Otra posible explicación acude a la naturaleza dinámica del problema, pues se necesita actualizar las soluciones constantemente debido a los cambios del entorno.

3.1.1. Soluciones exactas

La mayoría de los enfoques para soluciones exactas son basadas en métodos de programación mixta-entera o de programación dinámica. Dentro de los enfoques mediante **programación dinámica**, *Pasaraftis* [9] desarrolla uno de los primeros para abordar ALP, involucrando una pista de aterrizaje para luego extenderla a dos. En su trabajo, todos los aviones pueden aterrizar en tiempo 0 y no hay restricciones impuestas con respecto a los delays. Es más, se considera que el set de aviones puede ser particionado en un pequeño número de clases: aviones que pertenecen a la misma clase tienen similitudes en el proceso de planificación. Se investigan dos objetivos: **Minimizar el flujo de la pista de aterrizaje**, equivalente a minimizar el tiempo de aterrizaje del último avión en la secuencia, y **minimizar el costo total de demora de los pasajeros**, que depende de la clase del avión.
Por otro lado *Briskorn y Stolletz* [3] extienden este enfoque a múltiples pistas independientes, y utilizando las ventanas de tiempo para el aterrizaje mencionadas anteriormente. Bajo la suposición de que los aviones están particionados en clases y bajo la regla de las ventanas de tiempo de aterrizaje más temprano, probaron que dentro de cada clase, es óptimo planificar acorde a la regla FCFS. Luego, es modelado buscando el camino más corto en un grafo acíclico y luego resolverlo utilizando programación dinámica. De todas formas, la complejidad polinomial de este enfoque es exponencial en el número de clases de aviones: $O(RW^{R+1}n^{(R+1)W^2+R+1})$ donde R es el número de pistas, y n el número de avión. Para implementar este enfoque, adaptaron el modelo MIP a su función objetivo, el cual depende de cada clase de avión, para luego ser resuelto utilizando CPLEX, un software de optimización. A pesar de ello, existe otro enfoque que utiliza este modelo con algunas modificaciones: no permite los aterrizajes antes del tiempo objetivo y utiliza un criterio de dominancia que selecciona los estados para los cuales las pistas se liberan antes. Estas modificaciones permiten implementar eficientemente el algoritmo de programación dinámica.
Dentro de los enfoques utilizando **MIP**, se mencionó el de Beasley, el cual puede ser resuelto utilizando CPLEX para un número incremental de pistas (hasta cuatro pistas de aterrizaje), como también el enfoque utilizando el job-scheduling problem, el cual ha sido probado involucrando de dos hasta cinco pistas de aterrizaje, sin embargo, utilizando MIP se obtuvieron tiempos de cómputo altos.

3.1.2. Soluciones estocásticas

La naturaleza dinámica del problema ha motivado a estas soluciones que tienen tiempos de cómputo bajos y permiten actualizar las soluciones cuando un nuevo evento ocurre. Por lo tanto, muchos investigadores están a favor de utilizar estos enfoques heurísticos y metaheurísticos.
En la literatura previa al 2010, los algoritmos genéticos han sido los más utilizados. Recientemente, **Tabu Search y Simmulated Annealing** (Búsqueda Tabú y Recocido Simulado) han ganado bastante atención y se han convertido en las metaheurísticas más utilizadas, por lo que a continuación, se profundizará un poco más en ellas.

En la **búsqueda Tabú**, *Furini* [4] considera el problema en una sola pista de aterrizaje, donde involucra dos tipos de restricciones: la separación y la ventana de tiempos, con el objetivo de minimizar la demora ponderada total. Se propone un enfoque de horizonte rodante, que consiste en subdividir la instancia inicial en un set de sub-instancias, llamados chunks, que

siguen ciertas reglas. Luego, secuencialmente resolver cada instancia individualmente, ya sea utilizando un método MIP o TS (Tabu Search). En su TS propuesto, una solución candidata es definida mediante la permutación del set de aviones. Las soluciones vecinas son obtenidas al intercambiar dos posiciones de aviones, o cambiando un avion a una nueva posición si la nueva solución candidata no está prohibida por la lista tabú. El método TS continúa mejorando la solución candidata hasta un criterio de detención (un máximo número de iteraciones o tiempo límite).
Por otro lado, *Soykan y Rabadi* [2] proponen un enfoque más general al problema considerando múltiples pistas de aterrizaje independientes. Su enfoque de descompone en dos pasos principales:

1. Similar a la secuencia FCFS, calcula una solución inicial utilizando un enfoque greedy, llamado *Target Time First Greedy Algorithm* (TTFGA), que consiste en en realizar el atterizaje o despegue según su orden ascendente de tiempos objetivo, y en la pista en que la demora ponderadada es mínima.

2. Mejorar la solución inicial. Las soluciones candidatas son generados tanto intercambiando dos aviones en la misma pista, o en diferentes pistas, o eliminando/insertando un avión fuera/dentro de las secuencias. Cuando se encuentra un mejor valor de la función objetivo, se utiliza un mecanismo de aspiración para ignorar las restricciones tabú. Cuando no ocurre una mejora dentro de un cierto número de iteraciones, el algoritmo termina.

Los resultados con este enfoque tienen tiempos de cómputo muy cortos (menores que un segundo) y un gap con la solución óptima de un 10.15 %. Sin embargo, la brecha puede ser mayor para algunas instancias.

En **Simulated annealing**, *Hancerliogullari* [5], quien utilizó el modelo enfocado al job-scheduling problem, presentó una heurística SA para abordar su modelo, donde su initial guess es construido utilizando tres tipos de algoritmos greedy. El **Earliest Ready Time** (ERT) que consiste en aterrizar los aviones acorde a su orden de llega, en la pista de aterrizaje a la que puede acceder antes, el **Adapted Apparent Tardiness Cost with Separation and Ready Times** (AATCSR) y **Fast Priority Index** (FPI) inspirados de la literatura asociada al job-shop scheduling problem, donde aterriza cada avión según un índice de prioridad. Las soluciones candidatas en el SA se generan a partir de escoger aleatoriamente dos aviones e intercambiar sus posiciones, siempre que sus restricciones de ventana de tiempos no sean quebrantadas.

4. Modelo Matemático

4.1. Modelo estático

A continuación, se presenta el modelo estático propuesto por Beasley [8] que considera múltiples pistas de aterrizaje.

4.1.1. Variables

$x_i :=$ Tiempo de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$\alpha_i :=$ Cuán pronto el avión i aterriza antes de $T_i \quad \forall i \in 1, ..., P$

$\beta_i :=$ Cuán pronto el avión i aterriza después de $T_j \quad \forall i \in 1, ..., P$

$$\delta_i := \begin{cases} 1: & \text{Si el avión i aterriza antes que j} \quad \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j \\ 0: & \text{Otro caso} \end{cases}$$

$$y_{ir} := \begin{cases} 1: & \text{Si el avión i aterriza en la pista r} \quad \forall i \in 1, ..., P \ \forall r \in 1, ..., R \\ 0: & \text{En otro caso} \end{cases}$$

$$z_{ij} := \begin{cases} 1: & \text{Si el avión i y j aterrizan en la misma pista} \quad \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j \\ 0: & \text{En otro caso} \end{cases}$$

4.1.2. Parámetros

$R :=$ Número de pistas

$P :=$ Número de aviones

$E_i :=$ Tiempo más temprano de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$L_i :=$ Tiempo más tardío de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$T_i :=$ Tiempo objetivo de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$S_{ij} :=$ Tiempo de separación requerido (≥ 0) entre el aterrizaje del avión i y el aterrizaje del avión j donde i aterriza antes que j $\quad \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j$

$s_{ij} :=$ Tiempo de separación requerido (≥ 0) entre el aterrizaje del avión i y el aterrizaje del avión j donde i aterriza antes que j en distintas pistas $\quad \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j$

$g_i :=$ Costo de penalización (≥ 0) por unidad de tiempo de aterrizaje antes del tiempo objetivo T_i para el avión i $\quad \forall i \in 1, ..., P$

$h_i :=$ Costo de penalización (≥ 0) por unidad de tiempo de aterrizaje después del tiempo objetivo T_i para el avión i $\quad \forall i \in 1, ..., P$

Ocurre que para ciertos pares de aviones se puede decidir inmediatamente si uno aterriza antes que otro, por ejemplo, dos aviones i y j donde sus ventanas de tiempo son $[10, 50]$ y $[70, 110]$ respectivamente, es claro que el avión i debe aterrizar antes que j, sin embargo, no se sabe si la **restricción de tiempos de separación es satisfecha automáticamente**, pues puede que la separación entre aterrizaje de esos aviones sea $S_{ij} = 15$, caso en donde la restricción sería satisfecha, por otro lado, puede que $S_{ij} = 25$ y en ese caso no lo sería. Por lo tanto, se deben considerar tres sets:

- U: El set de pares de aviones (i,j) para los cuales no se sabe con certeza si el avión i aterriza antes que al avión j o no
- V: El set de pares de aviones (i,j) para los cuales se sabe que avión i aterriza antes que al avión j (pero que la restricción de separación de tiempos no es satisfecha automáticamente)
- W: El set de pares de aviones (i,j) para los cuales se sabe que avión i aterriza antes que al avión j (y la restricción de separación de tiempos es satisfecha automáticamente)

Se debe considerar los tiempos de separación S_{ij} y s_{ij}. Luego, los sets se definen como:

$$W := [(i,j) \mid L_i < E_j \ \wedge \ L_i + max(S_{ij}, s_{ij}) \leq E_j \ \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j]$$
$$V := [(i,j) \mid L_i < E_j \ \wedge \ L_i + max(S_{ij}, s_{ij}) > E_j \ \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j]$$
$$U := [(i,j] \mid \forall i \in 1, ..., P \ \forall j \in 1, ..., P \ i \neq j \ E_j \leq E_i \leq L_j \ \vee \ E_j \leq L_i \leq L_j \ \vee \ E_i \leq L_j \leq L_i]$$

La ventana de tiempo para el avión i es, entonces, $[E_i, L_i]$, donde $E_i \leq T_i \leq L_i$.

4.1.3. Restricciones

1. Esta restricción asegura que cada avión aterrice dentro de su ventana de tiempo.

$$E_i \leq x_i \leq L_i \forall i \in 1, ..., P$$

2. El avión i debe aterrizar antes que el avión j ($\delta ij = 1$) o el avión j debe aterrizar antes que el avión i ($\delta_{ji} = 1$)

$$\delta_{ij} + \delta_{ji} = 1 \forall i \in 1, ..., P; \forall j \in 1, ..., P; j > i$$

3. Considerando los sets definidos en la sección de parámetros, la siguiente restricción representa los sets de pares de aviones donde sabemos que el avión i aterriza antes que j

$$\delta_{ij} = 1 \quad \forall (i,j) \in W \cup V$$

4. Se necesita una restricción de separación para los pares de aviones en V, asegurando que el tiempo S_{ij} transcurra después del aterrizaje del avión i en x_i antes que el avión j pueda aterrizar en x_j para el caso de estar en la misma pista, y de manera análoga en caso de estar en pistas distintas.

$$x_j \geq x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) \quad \forall (i,j) \in V$$

5. Para la restricción de separación para los pares de aviones en U, en primer lugar, se considera M una constante positiva muy grande.

$$x_j \geq x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - M\delta_{ji} \quad \forall (i,j) \in U$$

   Se puede notar que si $\delta_{ji} = 0$, significa que el avión i aterrizó antes que j, asegurando la restricción de separación pues $x_j \geq x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij})$. Por otro lado, si $\delta_{ji} = 1$, $x_j \geq x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - M$, por lo tanto, $x_j \geq$ *algún número negativo*, por lo que la restricción queda inactiva.
   Se puede reemplazar M por $L_i + max(S_{ij}, s_{ij}) - E_j$ para la restricción final, puesto que debido al set U, significa que en la ventana de tiempo están, por ejemplo, las ventanas $[0, 10]$ y $[8, 10]$ para avión i y j respectivamente, y no se puede saber cuál aterrizará primero debido a que $L_i > E_j$. Entonces, en caso de $\delta_{ji} = 1$, $x_j \geq x_i - L_i + E_j$, y luego la restricción siempre se cumple puesto que x_j será mayor a $x_i -$ *algo* si el avión j aterriza antes.

$$x_j \geq x_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - (L_i + max(S_{ij}, s_{ij}) - E_j)\delta_{ji} \quad \forall (i,j) \in U$$

6. A continuación, se necesitan restricciones para relaciones las variables α_i, β_i y x_i, y necesarias para asegurar que se obtendrá una función objetivo lineal. La primera, asegura que el tiempo cuán pronto el avión i aterriza antes de su tiempo objetivo T_i sea mayor o igual a la diferencia entre el tiempo objetivo de i y su tiempo de aterrizaje. Esta restricción es

mayor o igual debido a que, en caso de que el avión aterrice antes, la restricción sería igual a $T_i - x_i$, pero en caso de que α_i sea 0, es decir, los casos donde no aterriza antes de T_i, significa que aterriza en T_i, y se cumpliría $0 = 0$, o aterriza después de T_i, donde $T_i - x_i$ sería negativo y $0 \geq T_i - x_i$.

$$\alpha_i \geq T_i - x_i \quad \forall i \in 1, ...P$$

7. En esta restricción se definen los valores que puede tomar α_i, puesto que como se mencionó antes, su valor será 0 en caso de aterrizar en T_i o después, y en caso contrario, el mayor valor que puede tomar es aterrizar en el tiempo más temprano E_i.

$$0 \leq \alpha_i \leq T_i - E_i \quad \forall i \in 1, ...P$$

8. Las siguientes restricciones relacionan de manera análoga las variables β_i con x_i

$$\beta_i \geq x_i - T_i \quad \forall i \in 1, ...P$$

9.

$$0 \leq \beta_i \leq L_i - T_i \quad \forall i \in 1, ...P$$

10. Esta restricción define el tiempo x_i, siendo el tiempo objetivo T_i menos cuán pronto o después aterrizó de su tiempo objetivo, dando como resultado su tiempo de aterrizaje x_i

$$x_i = T_i - \alpha_i + \beta_i \quad \forall i \in 1, ...P$$

11. Las siguientes restricciones se relacionan a la cantidad de pistas de aterrizaje. Esta asegura que cada avión aterrice en exactamente una pista de aterrizaje.

$$\sum_{r=1}^{R} y_{ir} = 1 \quad \forall i \in 1, ...P$$

12. Esta restricción asegura la simetría, es decir, si los aviones i y j aterrizan en la misma pista de aterrizaje, también lo hacen j e i.

$$z_{ij} = z_{ji} \quad \forall i \in 1, ...P \ \forall j \in 1, ..., P \ j > i$$

13. Esta restricción asegura que si hay alguna pista r para la cual y_{ir} y y_{jr} son 1, entonces fuerza a que z_{ij} sea 1. En caso de que sea 0, asegura que los aviones i y j no aterrizan en la misma pista, ya que si y_{ir} es 1, es decir, i aterriza en la pista r, y_{jr} solo puede tomar el valor 0 (no aterriza en la pista r) para cumplir que z_{ij} sea mayor.

$$z_{ij} \geq y_{ir} + y_{jr} - 1 \quad \forall i \in 1, ...P \ \forall j \in 1, ..., P \ j > i \ \forall r \in 1, ..., R$$

4.1.4. Función objetivo

Minimizar el costo total de la desviación de los aviones de su tiempo objetivo, puesto que ya se conoce el costo g_i asociado a aterrizar antes del tiempo objetivo, como también el costo h_i asociado a aterrizar después del tiempo objetivo, y mediante las variables α_i y β_i se sabe las unidades de tiempo en las que el avión i aterriza antes o después respectivamente.

$$mín = \sum_{i=1}^{P} (g_i \alpha_i + h_i \beta_i)$$

4.2. Modelo dinámico

En este modelo dinámico, se considerará una sola pista de aterrizaje, a diferencia del anterior. Anteriormente, se conocía con antelación los parámetros asociados tales como la cantidad de aviones, en cambio, en la versión dinámica, los controladores tienen conocimiento del avión entre 30 y 40 minutos antes de que lleguen a la pista, por lo que el número de aviones no se conoce. Por otro lado, en esta versión, la función objetivo consiste en minimizar los tiempos de separación entre aviones, sujetos a cumplir con los requerimientos de seguridad [7].

4.2.1. Variables

$$x_i := \text{Tiempo de aterrizaje del avión i} \quad \forall i \in 1, .., P$$

4.2.2. Parámetros

$P :=$ Número de aviones

$E_i :=$ Tiempo más temprano de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$L_i :=$ Tiempo más tardío de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$T_i :=$ Tiempo objetivo de aterrizaje para el avión i $\quad \forall i \in 1, ..., P$

$\delta_i^e :=$ Tolerancia permitida de aterrizaje antes del tiempo objetivo $\quad \forall i \in 1, ..., P$

$\delta_i^l :=$ Tolerancia permitida de aterrizaje después del tiempo objetivo $\quad \forall i \in 1, ..., P$

$ult_i :=$ Tiempo en el que el avión i debería aterrizar si no hay otros aviones que impidan el progreso en la pista. Determinado por el sistema de planificación de llegadas luego de que el avión entre en el rango del radar. Se asume que el avión no llega antes de este tiempo $\quad \forall i \in 1, ..., P$

$ts_i :=$ Máximo tiempo en el que puede aterrizar luego de $ult_i \quad \forall i \in 1, ..., P$

$e_i := max(E_i, ult_i)$

$l_i := min(L_i, ult_i + ts_i)$

$C :=$ Número de clases de aviones

$s_{bc} :=$ Mínima separación de tiempo cuando un avión de clase b aterriza antes que un avión de clase c $\forall b, c \in 1, ..., C$

$p_c :=$ Número de aviones en la clase c $\quad \forall c \in 1, ..., C$

$g_i :=$ Costo de penalización (≥ 0) por unidad de tiempo de aterrizaje antes del tiempo objetivo T_i para el avión i $\quad \forall i \in 1, ..., P$

$h_i :=$ Costo de penalización (≥ 0) por unidad de tiempo de aterrizaje después del tiempo objetivo T_i para el avión i $\quad \forall i \in 1, ..., P$

$v_i^l :=$ Costo por unidad de tiempo del combustible extra debido al retraso luego de ult_i

4.2.3. Restricciones

1. El tiempo de llegada de un avión debe estar entre el intervalo de ventanas de tiempo combinada por el tiempo más tardío y temprano junto con las desviaciones de ult_i

$$e_i \leq x_i \leq l_i \quad \forall i \in 1, .., P$$

2. Desigualdad triangular

$$s_{ab} + s_{bc} \geq s_{ac}$$

3. La cantidad de aviones debe ser igual a la sumatoria de los aviones en cada clase

$$P = \sum_{c=1}^{C} p_c$$

4. A continuación, se puede representar un avión con su número de avión y su clase de forma i, b donde el avión i pertenece a la clase b, o j, c, avión j que pertenece a la clase c. En esta restricción se manejan los tiempos de separación según clase, dependiendo de si un avión i aterriza antes o después de j, donde su tiempo de llegada x_i debe cumplir con la restricción de tiempos de separación.

$$x_{i,b} + s_{bc} \leq x_{j,c} \vee x_{j,c} + s_{cb} \leq x_{i,b}$$

4.2.4. Función objetivo

El principal objetivo es maximizar el throughput de la pista, lo que se traduce a minimizar el tiempo de aterrizaje del último avión en la planificación, es decir minimizar x_{max}

$$LT_{max} = max_i^{1,\dots,P} LT_i$$

En un entorno más realista, existe una gran probabilidad de que la última parte de la planificación cambie debido a la llegada de un nuevo avión, con el resultado de que solo la parte inicial de la planificación es implementada. Por lo tanto, se considera también la minimización del tiempo de aterrizaje promedio, buscando reducir cada tiempo de aterrizaje en vez de solo el último. La contribución total a la función objetivo es:

$$ALT = \sum_{j=1}^{n} \frac{LT_j}{n}$$

Luego, para cada avión i hay una ventana de tiempo $[T_i - \delta_i^e, T_i + \delta_i^l]$ dentro de la cual el avión debería idealmente aterrizar. Si $x_i < T_i - \delta_i^e$, existe una penalización por aterrizar antes, de forma similar al modelo anterior (solo que ahora se tiene una tolerancia al tiempo de aterrizaje): $g_i(T_i - \delta_i^e - x_i)$ y de forma análoga en caso de aterrizar después. Se obtiene la siguiente penalización total:

$$TW = \sum_{i}^{P} g_i max(T_i - \delta_i^e - x_i, 0) + \sum_{i}^{P} h_i max(T_i - \delta_i^l - x_i, 0)$$

Luego, existe un costo asociado a utilizar más combustible del necesario, además de que la reducción de quema de combustible ayuda a los objetivos del gobierno con respecto a las emisiones de CO_2. Así, otro objetivo es la minimización del combustible adicional utilizado para la programación de aterrizajes.

$$EF = \sum_{i=1}^{P} \upsilon_i^l max(x_i - ult_i, 0)$$

Finalmente, la **función objetivo** se obtiene como la suma ponderada de los objetivos mencionados antes individualmente, donde w_1, w_2, w_3 y w_4 son los pesos no-negativos asociados a cada objetivo. Se busca minimizar esta función objetivo.

$$mín = w_1 x_{max} + w_2 ALT + w_3 TW + w_4 EF$$

5. Representación

Para la siguiente implementación, se realizan las representaciones y descripciones considerando una simplificación del problema donde existe **una sola pista de aterrizaje**.

La representación en la construcción de las soluciones consiste en un arreglo de estructuras tipo avión, las cuales guardan la información de cada avión, es decir, su tiempo ideal de aterrizaje, el tiempo más temprano y tardío de aterrizaje, las penalizaciones por aterrizar antes o después, un arreglo que contiene la separación necesaria entre aviones, y el tiempo de aterrizaje X_i instanciado.

Al obtener una solución, la representación consiste en una estructura que incluye un arreglo del largo de la cantidad de aviones p que contiene la variable tiempo de aterrizaje X_i asignado a cada avión, y el costo total de la solución. Este costo se calcula antes de guardar la solución utilizando la representación anterior, aplicando las penalizaciones para calcular el costo total por aterrizar antes o después.

Dado que se trabaja con una técnica completa (Backtracking con Conflict-based backjumping), es necesario definir el dominio que se utilizará para instanciar las variables, el cual corresponde al intervalo de tiempo entre el tiempo más temprano de aterrizaje y tardío para avión. Al definir este dominio, se encapsula la restricción asociada a que el tiempo de aterrizaje debe estar entre estos valores. A pesar de ello, se propone ordenar este dominio según cuán alejado está del tiempo ideal de aterrizaje, de forma que se vayan instanciando antes los tiempos que menos penalización impliquen. Un ejemplo es, considerando como tiempo más temprano de aterrizaje 129, tiempo ideal de aterrizaje 155, y tiempo más tardío de aterrizaje 200, un orden del dominio para esta variable de la siguiente forma:

[155 154 156 153 157 152 158 151 159 150 160 149 161 148 162 147 163 146 164 145 165 144 166 143 167 142 168 141 169 140 170 139 171 138 172 137 173 136 174 135 175 134 176 133 177 132 178 131 179 130 180 129 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200]

El dominio ordenado para cada variable se guarda en un arreglo dentro de la estructura tipo avión.

Otra consideración al tener en cuenta al utilizar esta técnica es el orden de instanciación de las variables, puesto que existen aviones con una penalización mayor, y sería conveniente instanciar los mejores valores (dado por el dominio anterior) para estos aviones en primer lugar y de esa forma conseguir mejores soluciones en un inicio. Dado esto, se agrega a la representación de los aviones un parámetro que contiene el promedio de penalización, considerando la penalización por aterrizar antes y por aterrizar después. En el ejemplo a continuación, las variables se instanciarían en el siguiente orden: **Avión 3, avión 4, avión 5, avión 1, avión 2**.

Avión	Penalización g_i	Penalización h_i	Penalización promedio
1	10.00	10.00	10.00
2	10.00	10.00	10.00
3	30.00	30.00	30.00
4	30.00	30.00	30.00
5	30.00	30.00	30.00

Cuadro 1: Ejemplo de penalizaciones para aviones

La idea inicial para este problema fue la instanciación de las variables en el orden dado, y el orden de dominio del tiempo más temprano de aterrizaje al más tardío, sin embargo, dado que las primeras instanciaciones completas (soluciones) ocurrían para tiempos alejados (los más tempranos) del tiempo ideal de aterrizaje de los aviones, el costo era mayor, y luego se desactivaba el salto inteligente, para utilizar salto cronológico, demorando más en avanzar hacia soluciones con tiempos más cercanos al tiempo ideal. La idea final de orden de instanciación y de

dominio, en cambio, brinda en un inicio una solución con un costo mucho menor a las soluciones iniciales de la idea anterior, debido a que es una solución factible con tiempos de aterrizaje más cercanos al ideal. Un ejemplo considerando esta heurística es que si la instanciación donde todos los aviones aterricen en su tiempo de aterrizaje ideal es factible, esta será la primera solución en entregar.

6. Descripción del algoritmo

El algoritmo utilizado fue **Backtracking con Conflict-based backjumping** en C.

Implementando la representación mencionada anteriormente, se utilizan las estructuras en C, guardando los datos de los aviones y del problema extraídos del archivo txt con el formato de entrada dado, en un arreglo de estructuras tipo avión. Para utilizar CBJ, es necesario guardar el conjunto de conflictos para cada variable, por lo que, se agrega a la estructura tipo avión una lista enlazada para ir agregando las variables con las que se tenga conflicto.

Al tener el arreglo de aviones, se aplican las modificaciones mencionadas anteriormente con respecto al orden de instanciación de variables y dominio, ordenando el arreglo de aviones según orden de instanciación de variable y agregando a cada avión un arreglo con el dominio a utilizar para la instanciación ordenado como se mencionó en la representación.

Al tener listas las representaciones, se ejecuta CBJ, comenzando desde el avión de posición 0 del arreglo de aviones. La forma de implementación fue utilizar Backtracking de forma recursiva, de modo que se aplica una instanciación, y luego se llama nuevamente a la función CBJ para instanciar la siguiente variable.

A continuación, se presenta un pseudo-código para las dos funciones claves en la implementación del algoritmo.

Algorithm 1 restricciones

Require: avión, posición variable, aviones
Ensure: bool, True o False
  for i = 0; i < posición variable; i++ **do**
    if (avión.xi >= aviones[i].xi && avión.xi < aviones[i].xi + separacion con avión i) || (avión.xi <= aviones[i].xi && aviones[i].xi < avión.xi + separacion con avión i) **then**
      se agrega posición i de variable a lista de conflictos de avión
      return $False$
    end if
  end for
  return $True$

El **algoritmo 1** trata las restricciones del problema, donde las principales consisten en cumplir con la separación de tiempos entre cada par de aviones. En este algoritmo, se chequea las restricciones de la nueva instanciación de un avión con todos los aviones instanciados anteriormente y se abordan los siguientes casos:

1. Cuando el nuevo avión instanciado (avión) aterriza después que el avión de posición i: la restricción de separación no se cumple cuando el avión aterriza antes que la suma del aterrizaje del avión i más la separación exigida entre estos dos aviones.

2. Cuando el nuevo avión instanciado (avión) aterriza antes que el avión de posición i: la restricción de separación no se cumple cuando el avión i aterriza antes que la suma del aterrizaje del avión más la separación exigida entre estos dos aviones.

3. Dado que la separación entre cada par de aviones debe ser mayor a 0, si el tiempo de aterrizaje para un par de aviones es el mismo, las restricciones tampoco se cumplen.

En caso de que alguno de los casos anteriores se cumpla, pues son los casos donde las restricciones son quebrantadas, el algoritmo retorna $False$ y se agrega la posición de la primera variable con la que se encontró conflicto, pues la primera encontrada será la más prematuramente instanciada, y en caso contrario, se retorna $True$, pues las restricciones se cumplen y es una instanciación consistente. Además, de la forma que se presentan las restricciones, también se encapsula la restricción 2 mencionada en el modelo estático donde el avión i debe aterrizar antes que el avión j o viceversa.

Cabe mencionar que se hace distinción entre el avión como variable y su posición como variable dentro del arreglo de aviones, dado la representación mencionada, donde la variable del avión (por ejemplo, avion 1, avion 2, avion 3) no coincide con su posición en la instanciación (dado por el arreglo de aviones).

Para el **algoritmo 2** que trata el Backtracking con CBJ, se comienza la llamada a esta función desde el avión de posición 0, considerado como el nodo raíz. Para todos los nodos después del nodo raíz, se verifican las restricciones, indicando si la instanciación es consistente o no.

En caso de que la instanciación de los aviones sea consistente, se sigue con la instanciación del avión en la posición variable, modificando su valor de X_i, donde luego se vuelve a llamar recursivamente a la función CBJ con el objetivo de instanciar el avión de posición variable + 1. Se puede notar como las restricciones se verifican para la instanciación del avión en la posición variable - 1, esto se debe a que primero se instancia un avión y luego se llama a CBJ para instanciar al *siguiente avión*, por lo que es necesario verificar si la instanciación del *avión* es consistente o no para poder seguir con la instanciación del *siguiente avión*. En caso de que sea inconsistente, esta llamada termina (avión de posición variable + 1), dando a paso a seguir con los demás valores del dominio del *avión*.

El término de parada para la recursión ocurre cuando todas las variables han sido instanciadas, es decir, de 0 a p-1 (en la llamada con posición p donde no hay más variables por instanciar) y esta instanciación es consistente, llegando a una solución. En este caso, se actualiza la mejor solución si el costo de la solución actual es menor, además de desactivar el salto inteligente, para poder encontrar más soluciones.

Luego de la llamada recursiva, existen distintos casos que se abordan.

1. Si el salto CBJ está desactivado y se llega al nodo raíz, se vuelve a activar el salto inteligente.

2. Si el salto CBJ está desactivado y no es nodo raíz, se reinicia la lista de conflictos para esa variable.

3. Si el salto CBJ está activado, hubo un fallo y la variable actual no es el nodo al que se realiza el salto, se reinicia su lista de conflictos.

4. Si el salto CBJ está activado, hubo un fallo y la variable actual es el nodo al que se realiza el salto, se reinicia el fallo y se sigue con la instanciación de esa variable.

Finalmente, cada avión tiene su variable local de fallo, la cuál se activa luego de instanciar todos sus valores de dominio donde el salto CBJ sigue activado, puesto que al intentar instanciar todos los valores de su dominio, no se consigue una instanciación consistente, produciéndose un **fallo**. Al ocurrir esta situación, se actualiza el salto CBJ informando de que hubo un fallo y debe saltar, escogiendo la variable de salto al máximo valor de la lista de conflictos de la variable (esta lista guarda las posiciones de la variable donde hubo conflicto, por lo que al saltar al máximo valor, salta a la más recientemente instanciada).

Se ha mencionado que hay información asociada al salto CBJ a la cual pueden acceder todas las variables, por ejemplo, si el salto inteligente está activado, si hubo un fallo, y a la variable a la que se debe saltar. Esta información se guarda en otra estructura de C con nombre $salto_{CBJ}$.

Algorithm 2 CBJ

```
Require: aviones, posición variable, p, mejor solución
  consistente ← True
  fallo ← False
  if no es nodo raíz then
      consistente ← restricciones(aviones[posición variable − 1], posición variable −
  1, aviones)
  end if
  if posición variable == p && consistente then
      if costo solución actual < costo mejor solución then
          mejor solución ← solución actual
      end if
      desactivar salto CBJ
      return
  end if
  if consistente then
      for dominio de variable do  /*Se instancia con un valor de dominio*/
          aviones[posición variable].xi ← dominio
          CBJ(aviones, posición variable + 1, p, mejor solución)
          if salto CBJ desactivado && nodo raíz then
              activar salto CBJ
          else if salto CBJ desactivado then
              reiniciar lista de conflictos de variable
          else if salto CBJ activado && hubo fallo then
              if no es nodo de salto then
                  reiniciar lista de conflictos de variable
                  return
              else if es nodo de salto then
                  reiniciar fallo
              end if
          end if
      end for
      if salto CBJ activado && no es nodo raíz then
          fallo ← True
      end if
  end if
  if fallo then
      actualizar que hubo un fallo
      salto CBJ a max(aviones[posición variable].conflictos)
  end if
```

7. Experimentos

En los siguientes experimentos, se ejecutará el programa para cinco instancias distintas, donde cada instancia se ejecutará dos veces, una con el orden de instanciación de variables dado y con el orden de dominio de tiempo más temprano a más tardío, mientras que la otra ejecución será con el orden de instanciación de variables según su penalización promedio y orden de dominio según alejamiento del tiempo objetivo, tal como se mencionó en la sección anterior. Esto se realiza con el objetivo de verificar cómo las heurísticas de selección de variables y valores pueden encontrar soluciones de mejor calidad en menos tiempo, dado que el uso de una técnica completa por su naturaleza determinista debe escoger la mejor solución de todas,

siendo necesario explorar todas las soluciones, siendo muy costoso en tiempo de ejecución, e incluso aumentándolo con la cantidad de aviones y el rango de tiempos de aterrizaje que tiene cada avión, pues debe instanciar una mayor cantidad de valores. Más aún, si la ventana de tiempos de aterrizaje de cada avión es grande, se puede llegar rápidamente a una solución, donde CBJ desactiva su salto inteligente, transformándose en salto cronológico, es decir, Backtracking, volviéndose más costoso pues el salto inteligente se vuelva a activar solo al llegar a la primera variable instanciada, el nodo raíz en el árbol de búsqueda asociado. Por ello, el uso de heurísticas podría ser apropiado en caso de que ejecutar el programa hasta que la técnica encuentre todas las soluciones no sea posible debido al tiempo empleado, una situación probable que ocurra dado el problema estudiado, a la gran cantidad de datos en los aeropuertos y a la necesidad de planificar constantemente el aterrizaje de los aviones.

La calidad de las soluciones será determinada según la función objetivo, el costo total que implica aquella instanciación de tiempos de aterrizajes.

Debido a lo anterior, cada ejecución se limita a 5 minutos, quedándose con la mejor solución encontrada hasta ese momento. La ejecución se realiza en el sistema operativo Linux, específicamente en Ubuntu 18.04.2.

Además, se comparará la cantidad de instanciaciones de variables realizadas en cada experimento, y particularmente la cantidad de saltos realizados en la primera instancia.

El formato de los archivos de prueba utilizados es el siguiente:

Número de aviones
p
Datos de aterrizaje del avión
$E_i\ T_i\ L_i\ g_i\ h_i$
Tiempo de separación entre pares de aviones
$S_{i1}\ S_{i2}\ ...\ S_{ii}\ ...\ S_{ip}$

La primera instancia *airline1.txt* contiene 10 aviones, con los siguientes datos:

10
129 155 559 10.00 10.00
99999 3 15 15 15 15 15 15 15 15
195 258 744 10.00 10.00
3 99999 15 15 15 15 15 15 15 15
89 98 510 30.00 30.00
15 15 99999 8 8 8 8 8 8 8
96 106 521 30.00 30.00
15 15 8 99999 8 8 8 8 8 8
10 123 555 30.00 30.00
15 15 8 8 99999 8 8 8 8 8
120 135 576 30.00 30.00
15 15 8 8 8 99999 8 8 8 8
124 138 577 30.00 30.00
15 15 8 8 8 8 99999 8 8 8
126 140 573 30.00 30.00
15 15 8 8 8 8 8 99999 8 8
135 150 591 30.00 30.00
15 15 8 8 8 8 8 8 99999 8
160 180 657 30.00 30.00
15 15 8 8 8 8 8 8 8 99999

Cabe destacar que el tiempo de separación S_{ii}, es decir de un avión consigo mismo se considera como $S_{ii} = 99999$. La segunda instancia a utilizar es *airline2.txt*, la cual contiene 15 aviones. La tercera instancia a utilizar es *airline3.txt*, que contiene 20 aviones, mientras que la cuarta *airline7.txt* presenta 44 aviones. Además para verificar el funcionamiento con más datos, se utiliza la quinta instancia *airline11.txt* con 200 aviones.

8. Resultados

A continuación, se presenta una tabla con los resultados obtenidos de los experimentos realizados con las cinco instancias mencionadas anteriormente, donde la propuesta 1 corresponde a la propuesta inicial con el orden de instanciación dado por las variables y su dominio del tiempo más temprano de aterrizaje al más tardío, mientras que la propuesta 2 es la propuesta final implementada con orden de instanciación según penalización promedio del más costo al más barato, y con orden de elección de valores del dominio según diferencia de tiempo con el tiempo de aterrizaje ideal para el avión, instanciando primero los tiempos más cercanos al ideal.

Instancia	Costo propuesta 1	Costo propuesta 2	Instanciaciones propuesta 1	Instanciaciones propuesta 2
airline1	3650	1150	521884989	492506962
airline2	8420	1720	448881681	443895932
airline3	21960	2090	446333049	424358738
airline7	3163	0	322892922	44
airline11	77103.335938	30704.789062	94756696	272325763

Cuadro 2: Resultados de los experimentos realizados con cinco instancias.

Como se puede notar, las soluciones con la propuesta final entregan un costo considerablemente menor, e incluso, la mayoría presenta una menor cantidad de instanciaciones.

La diferencia, como se explicó anteriormente, radica en que instanciar las variables con los valores de su dominio que estén más cerca del tiempo ideal de aterrizaje, incluyéndolo, permiten disminuir el costo por alejarse del tiempo ideal, además de que si se instancian las variables que tengan una mayor penalización promedio en primer lugar, se da prioridad a que estos aviones aterricen en su tiempo ideal o cerca de este, puesto que al tratarse de Backtracking con Conflict-based backjumping, al llegar a una solución, se desactiva el salto inteligente convirtiéndose en Backtracking, donde se tendrá que revisar cada elemento del dominio de cada variable hasta llegar a la raíz donde se vuelve a activar el salto. Considerando la ventana de tiempo que tiene cada avión, el llegar hasta el nodo raíz para volver a activar el salto inteligente conlleva un tiempo considerable, demorándo en poder cambiar el valor del dominio de las primeras variables instanciadas, de forma que si se mantiene estas variables en su tiempo ideal o cerca de él, se puede llegar a soluciones que disminuyan el costo considerablemente pues las variables que tienen un mayor costo por alejarse del tiempo de aterrizaje ideal, se encuentran en o cerca de este.

Se presenta el resultado obtenido para la instancia *airline1.txt* utilizando las dos propuestas.

```
Costo: 3650.000000
Tiempo total de ejecución: 300.000001 [s]
Instanciaciones totales en la ejecución: 521884989
Tiempo avión 1: 129
Tiempo avión 2: 195
Tiempo avión 3: 89
Tiempo avión 4: 97
Tiempo avión 5: 110
Tiempo avión 6: 144
Tiempo avión 7: 152
Tiempo avión 8: 160
Tiempo avión 9: 168
Tiempo avión 10: 180
```

```
Costo: 1150.000000
Tiempo total de ejecución: 300.000001 [s]
Instanciaciones totales en la ejecución: 492506962
Tiempo avión 3: 98
Tiempo avión 4: 106
Tiempo avión 5: 123
Tiempo avión 6: 135
Tiempo avión 7: 143
Tiempo avión 8: 151
Tiempo avión 9: 159
Tiempo avión 10: 180
Tiempo avión 1: 195
Tiempo avión 2: 258
```

Figura 2: Resultados obtenidos para la instancia *airline1*, utilizando las dos propuestas, siendo la primera imagen la propuesta inicial y la segunda la propuesta final implementada.

Se puede notar como en la segunda imagen, la instanciación de los aviones se hace en el orden 3, 4, 5, 6, 7, 8, 9, 10, 1, 2 debido a que los aviones del 3 al 10 tienen una penalización mayor, de 30 por alejarse del tiempo de aterrizaje, a diferencia de los aviones 1 y 2 que tienen penalización 10. Incluso, los aviones 3, 4, 5, 6 se encuentran en su tiempo ideal de aterrizaje.

Además, para esta instancia se presenta los saltos que realizó el algoritmo, los cuales fueron **32515** para la propuesta inicial, mientras que **32558** para la propuesta final.

Es importante señalar que si bien la solución encontrada luego de cinco minutos utilizando la heurística mencionada es de mejor calidad que la inicial, no significa que sea la mejor solución factible del espacio de búsqueda, aquello solo se puede verificar ejecutando la técnica completa hasta explorar todas las soluciones.

A pesar de lo anterior, el salto CBJ presenta la gran ventaja de que al mantener un conjunto de conflictos para cada variable, en caso de que ninguna instanciación a una variable sea factible, se salta a la más recientemente instanciada dentro del conjunto de conflictos para seguir instanciando desde ahí, evitando los casos que no serán factibles debido a una restricción anterior. Esto se puede notar en cómo hubo una gran cantidad de saltos en esta primera instancia, es decir, se encontraron varios fallos, y los saltos permitieron llegar a una solución de forma más rápida y con menos instanciaciones.

Los resultados de la instancia 7 *airline7* muestran cómo, en caso de que la instanciación con tiempos ideales de aterrizajes sea factible, esta será la solución entregada con costo 0 utilizando la heurística seleccionada, pues los tiempos ideales de aterrizaje son los primeros que se intentan instanciar para cada variable.

```
Costo: 3163.000000
Tiempo total de ejecución: 300.000001 [s]
Instanciaciones totales en la ejecución: 322892922
Tiempo avión 1: 0
Tiempo avión 2: 96
Tiempo avión 3: 168
Tiempo avión 4: 248
Tiempo avión 5: 328
Tiempo avión 6: 528
Tiempo avión 7: 624
Tiempo avión 8: 720
Tiempo avión 9: 792
Tiempo avión 10: 992
Tiempo avión 11: 1064
Tiempo avión 12: 1264
Tiempo avión 13: 1336
Tiempo avión 14: 1416
Tiempo avión 15: 1616
Tiempo avión 16: 1688
Tiempo avión 17: 1768
Tiempo avión 18: 1968
Tiempo avión 19: 2040
Tiempo avión 20: 2120
Tiempo avión 21: 2320
Tiempo avión 22: 2416
Tiempo avión 23: 2488
Tiempo avión 24: 2688
Tiempo avión 25: 2760
Tiempo avión 26: 2840
Tiempo avión 27: 2920
Tiempo avión 28: 3000
Tiempo avión 29: 3200
Tiempo avión 30: 3296
Tiempo avión 31: 3368
Tiempo avión 32: 3568
Tiempo avión 33: 3640
Tiempo avión 34: 3840
Tiempo avión 35: 3936
Tiempo avión 36: 4008
Tiempo avión 37: 4088
Tiempo avión 38: 4168
Tiempo avión 39: 4368
Tiempo avión 40: 4464
Tiempo avión 41: 4560
Tiempo avión 42: 4656
Tiempo avión 43: 4752
Tiempo avión 44: 4927
```

```
Costo: 0.000000
Tiempo total de ejecución: 0.000041 [s]
Instanciaciones totales en la ejecución: 44
Tiempo avión 1: 0
Tiempo avión 2: 137
Tiempo avión 6: 503
Tiempo avión 7: 626
Tiempo avión 8: 727
Tiempo avión 10: 967
Tiempo avión 12: 1239
Tiempo avión 15: 1591
Tiempo avión 18: 1943
Tiempo avión 21: 2295
Tiempo avión 22: 2391
Tiempo avión 24: 2663
Tiempo avión 29: 3175
Tiempo avión 30: 3271
Tiempo avión 32: 3543
Tiempo avión 34: 3815
Tiempo avión 35: 3911
Tiempo avión 39: 4343
Tiempo avión 40: 4439
Tiempo avión 41: 4535
Tiempo avión 42: 4631
Tiempo avión 43: 4727
Tiempo avión 23: 2591
Tiempo avión 4: 351
Tiempo avión 25: 2863
Tiempo avión 26: 2943
Tiempo avión 27: 3023
Tiempo avión 28: 3103
Tiempo avión 13: 1439
Tiempo avión 14: 1519
Tiempo avión 31: 3471
Tiempo avión 5: 431
Tiempo avión 33: 3743
Tiempo avión 16: 1791
Tiempo avión 17: 1871
Tiempo avión 36: 4111
Tiempo avión 37: 4191
Tiempo avión 38: 4271
Tiempo avión 9: 895
Tiempo avión 19: 2143
Tiempo avión 20: 2223
Tiempo avión 3: 271
Tiempo avión 11: 1167
Tiempo avión 44: 4927
```

Figura 3: Resultados obtenidos para la instancia *airline7*, utilizando las dos propuestas, siendo la primera imagen la propuesta inicial y la segunda la propuesta final implementada.

A continuación, se presenta los resultados para las demás instancias, exceptuando la 11 debido a la gran cantidad de aviones.

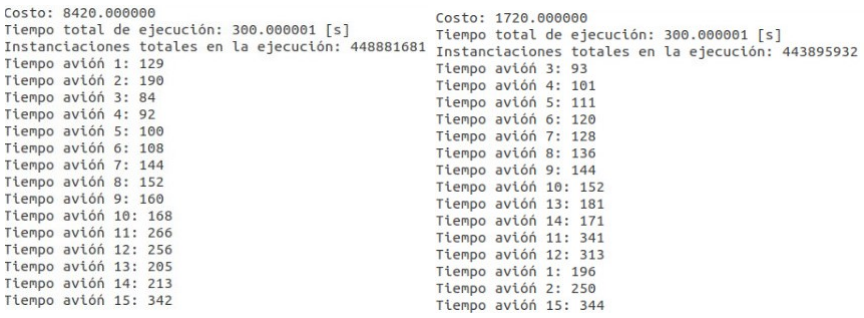

Figura 4: Resultados obtenidos para la instancia *airline2*, utilizando las dos propuestas, siendo la primera imagen la propuesta inicial y la segunda la propuesta final implementada.

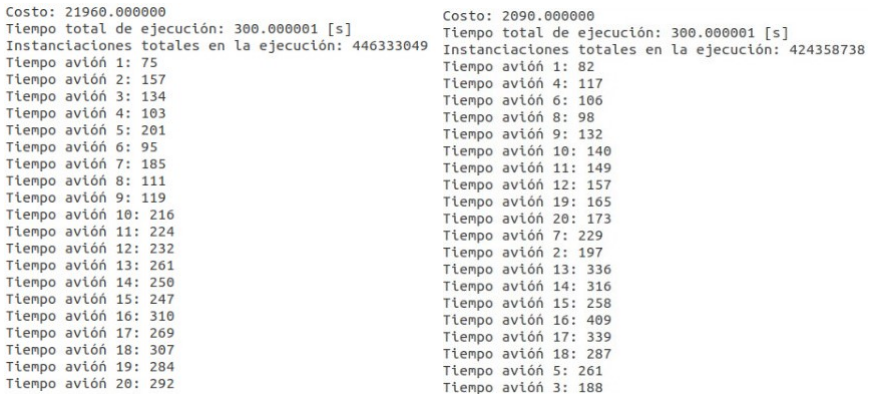

Figura 5: Resultados obtenidos para la instancia *airline3*, utilizando las dos propuestas, siendo la primera imagen la propuesta inicial y la segunda la propuesta final implementada.

9. Conclusiones

Se han propuesto distintos enfoques junto a varios métodos de solución asociado al *Aircraft Landing Scheduling Problem*, donde la formulación del problema presenta variantes, tales como: planificar los aterrizajes, despegues, o los dos, o si estamos frente a un aeropuerto con una pista de aterrizaje, o múltiples pistas. También, es importante tener en cuenta si se considerará un modelo estático, donde los datos se conocen con anticipación, o uno dinámico, donde el entorno está en constante cambio. Se debe cumplir con varias restricciones asociadas principalmente a la seguridad, pues existe una variedad de aviones volando, donde algunos provocan más turbulencia que otros, lo que se debe manejar utilizando tiempos de separación entre pares de aviones.

La elección del modelo dependerá principalmente del aeropuerto que hará uso de el, definiendo sus principales objetivos y la arquitectura de su aeropuerto, como también a qué tantos cambios en el entorno se enfrenta, para elegir entre un modelo estático o dinámico. Las principales limitaciones que han tenido estos modelos ha sido manejar la explosión combinatorial debido a la cantidad de pistas de aterrizaje y clases de aviones, puesto que a mayor cantidad, aumenta la complejidad y los tiempos de cómputo.

La planificación de las pistas de aterrizaje es un problema actual de la vida real, más aún con la creciente demanda que se debe cumplir con más aviones y mayor frecuencia de vuelos, provocando un aumento de los datos para el problema cada vez mayor. Frente a esto, se necesitan algoritmos para planificación que provean un mejor tiempo de ejecución, tanto por el aumento de datos como también por la necesidad de una respuesta rápida, por ello, los algoritmos que

proveen soluciones exactas se han quedado atrás, pues la tendencia ha sido optar por métodos estocásticos que brinden una solución cercana a la óptima para esta planificación en un tiempo reducido. Las técnicas más populares en este último tiempo han sido la **Búsqueda Tabú** y **Recocido Simulado**.

Aún existe trabajo que realizar en mejorar los distintos modelos que existen para este problema que adopten un enfoque más realista, y seguir investigando en los métodos de resolución que puedan hacer frente a la cantidad de datos crecientes presente en los aeropuertos. Es importante seguir trabajando en este problema, pues la demanda en los aeropuertos es cada vez mayor, y se necesitará mejores formas de planificar sus vuelos con el fin de ahorrar costos y mejorar el flujo. Se propone seguir la tendencia al uso de métodos estocásticos, promoviendo la investigación con distintos algoritmos en esta área, con el fin de poder hacer frente a la creciente cantidad de datos de los aeropuertos. También, se propone el uso de un modelo con múltiples pistas de aterrizaje que combine la propuesta estática como dinámica, pues, existen varios datos en los aeropuertos que se conocen con anticipación que servirían para proponer una planificación inicial, para luego, poder hacer frente a los cambios en el entorno, tales como retraso de vuelos o situaciones climáticas, utilizando un modelo dinámico, y aprovechando el uso de los métodos estocásticos para actualizar las planificaciones en poco tiempo.

La propuesta de solución utilizando Backtracking junto a Conflict-based backjumping incluyen escoger una adecuada representación de las soluciones, como también definir un orden de instanciación para las variables y un orden para la elección de valores de los dominios. Considerando el tiempo que involucra utilizar una técnica completa, se decide utilizar un orden de instanciación y elección de valores que permita encontrar soluciones de mejor calidad en un inicio al escoger los valores del dominio que menos penalización tengan dado que están en su tiempo ideal de aterrizaje, o cerca de él, además de instanciar en primer lugar las variables que tengan mayor costo, comenzando con las soluciones donde estas variables estén más cerca de su tiempo ideal de aterrizaje. Utilizando esta heurística puede ocurrir que si la instanciación donde todos los aviones aterricen en su tiempo de aterrizaje ideal es factible, esta será la primera solución en entregar.

El utilizar técnicas completas para el Aircraft Landing Scheduling Problem que brinden la mejor solución ha sido un desafío para este problema, debido a la explosión combinatorial. Concretamente en esta propuesta ocurre al aumentar la cantidad de aviones y el tamaño del dominio de los aviones, es decir, su rango posible de tiempos de aterrizaje, por lo que para encontrar la mejor solución se necesitan grandes tiempos de cómputo, además de mayor memoria en el caso de Conflict-based backjumping para almacenar el conjunto de conflictos para cada variable y visualizar los culpables de los conflictos.

En concreto para este problema, una desventaja observada es que el rango de tiempo en que los aviones pueden aterrizar (desde el tiempo más temprano al más tardío) es grande, por lo que existen varios valores dentro del dominio de cada variable a intentar, y si bien CBJ permite realizar saltos y disminuir la cantidad de instanciaciones con respecto a Backtracking, debido al gran rango de tiempos que tienen las variables, es más probable llegar a una solución factible donde las restricciones se cumplan, provocando que el salto inteligente se desactive y se convierta en cronológico para seguir encontrando soluciones. Puesto que el salto inteligente se vuelve activar solo al llegar al nodo raíz, se debe probar todos los valores para las variables, implicando un gran tiempo de cómputo tal como Backtracking, siendo difícil encontrar todas las soluciones para escoger la mejor entre ellas.

Otra desventaja en este problema es la cantidad de tiempo involucrada en verificar las restricciones, puesto que, en este problema cada avión debe cumplir una restricción de tiempo de separación con todos los demás aviones.

El beneficio de la heurística utilizada se pudo verificar con los experimentos utilizando cinco instancias, donde se ejecutó la técnica durante cinco minutos puesto que revisar todas las soluciones como lo hacen las técnicas completas es costoso en tiempo y computacionalmente. En los resultados se pudo notar cómo el costo de las primeras soluciones encontradas era considera-

blemente menor utilizando heurísticas de elección de variables y valores a que si se realizaba la instanciación de las variables en el orden dado y los valores del dominio de tiempo más temprano a más tardío. Además, se pudo notar que el salto inteligente fue beneficioso en este caso, pues hubo una gran cantidad de saltos, es decir, varios fallos los cuales fueron evitados utilizando Conflict-based backjumping.

Dado lo mencionado anteriormente, para trabajo futuro utilizando la técnica completa de Backtracking con Conflict-based backjumping, se propone mejorar las heurísticas de selección de variables y valores en el dominio. Por ejemplo, puede que un avión aterrizando después de su tiempo de aterrizaje ideal tenga un mayor costo que aterrizar antes, por lo que el orden de elección de valores del dominio se podría ordenar según el costo que tenga alejarse del tiempo ideal; por ejemplo, si alejarse dos unidades de tiempo antes del tiempo de aterrizaje sea más barato que alejarse una unidad del tiempo de aterrizaje después. Además, para verificar el comportamiento de esta nueva heurística, se propone realizar los mismos experimentos de comparación presentados.

Considerando el gran tiempo que implica utilizar una técnica completa, aplicar o experimentar con distintas heurísticas de elección de variables y valores permitiría encontrar soluciones de mejor calidad en una menor cantidad de tiempo. De esta forma, si se desea detener la ejecución de la técnica y quedarse con la solución encontrada hasta el momento, es más probable encontrar una solución de buena calidad.

Referencias

[1] William Raillant-Clark Anthony Philbin. Crecimiento sostenido del tráfico de pasajeros y demanda moderada de servicios de carga aérea en 2018. *Sala de prensa OACI*, 2018. `https://www.icao.int/Newsroom/Pages/ES/Solid-passenger-traffic-growth-and-moderate-air-cargo-demand-in-2018.aspx` [último acceso 26 Oct 2021].

[2] Ghaith Rabadi Bulent Soykan. A tabu search algorithm for the multiple runway aircraft scheduling problem. *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling. International Series in Operations Research Management Science*, 236:165–186, 2016.

[3] Raik Stolletz Dirk Briskorn. Aircraft landing problems with aircraft classes. *Journal of Scheduling*, 17(1):31–45, 2014.

[4] C.A. Persiani-Paolo Toth Fabio Furini, Martin Philip Kidd. Improved rolling horizon approaches to the aircraft sequencing problem. *Journal of Scheduling*, 18:435–447, 2015.

[5] Ameer H. Al-Salem Mohamed Kharbeche Gulsah Hancerliogullari, Ghaith Rabadi. Greedy algorithms and metaheuristics for a multiple runway combined arrivaldeparture aircraft sequencing problem. *Journal of Air Transport Management*, 32:39–48, 2013.

[6] Anita Berthier Hicham Ayoun. Aci world publishes annual world airport traffic report. *Airports Council International: Press releases*, 2018. `https://aci.aero/2018/09/20/aci-world-publishes-annual-world-airport-traffic-report/` [último acceso 26 Oct 2021].

[7] Y. M. Sharaiha J. E. Beasley, M. Krishnamoorthy and D. Abramson. Displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society*, 55(1):54–56, 2000.

[8] Y. M. Sharaiha J. E. Beasley, M. Krishnamoorthy and D. Abramson. Scheduling aircraft landings - the static case. *Transportation science*, 34(2):2–23, 2000.

[9] Harilaos N Psaraftis. A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(2):1347–1359, 1980.

[10] J. Desai R. Prakash, R. Piplani. An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput. *Transportation Research Part C Emerging Technologies*, 95, 2018.

[11] Marcel Mongeau Xavier Olive Emmanuel Rachelson Sana Ikli, Catherine Mancel. The aircraft runway scheduling problem: A survey. *Computers and Operations Research*, 132(1):4–39, 2021.