



MÁSTER UNIVERSITARIO EN INGENIERÍA
INDUSTRIAL

TRABAJO FIN DE MÁSTER

Diseño y validación de un modelo QUBO-QAOA
para la recuperación de aeronaves
ante disrupciones

Autor: Jorge García-Parreño Veguillas

Director: Pablo Díez Valle

Madrid

Junio 2026

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título

Diseño y validación de un modelo QUBO-QAOA para la recuperación de aeronaves ante disrupciones

en la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico **2025/2026** es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

☒ He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): «La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente». En concreto, la Inteligencia Artificial ha sido empleada para:

Investigación bibliográfica inicial y planificación de la memoria.

Autorización para la entrega del Proyecto

Firmado (alumno):



Fdo.: Jorge García-Parreño Veguillas

Fecha: 19/06/2026

El Director del Proyecto

Firmado por DIEZ VALLE PABLO - ***5708** el día 20/06/2026 con un certificado emitido por AC FNMT Usuarios

Fdo.: Pablo Díez Valle

Fecha: _____



MÁSTER UNIVERSITARIO EN INGENIERÍA
INDUSTRIAL

TRABAJO FIN DE MÁSTER

Diseño y validación de un modelo QUBO-QAOA
para la recuperación de aeronaves
ante disrupciones

Autor: Jorge García-Parreño Veguillas

Director: Pablo Díez Valle

Madrid

Junio 2026

Resumen

La gestión de interrupciones en la operación aérea constituye un problema de elevada complejidad combinatoria, especialmente cuando una avería o indisponibilidad de aeronave obliga a reprogramar vuelos en ventanas temporales reducidas. Este trabajo estudia la aplicación de técnicas de computación cuántica variacional al problema de recuperación de aeronaves, mediante el diseño y validación de un modelo QUBO-QAOA aplicado a una versión simplificada del problema.

En primer lugar, se analiza el contexto operativo de las operaciones irregulares en aerolíneas y se revisan los principales enfoques clásicos empleados en la recuperación de recursos, incluyendo métodos exactos, heurísticos y formulaciones basadas en redes espacio-temporales. A partir de este marco, se formula un modelo de recuperación de flota basado en variables binarias de asignación, retraso y cancelación, incorporando restricciones de unicidad y continuidad temporal. El modelo se expresa inicialmente como un problema de optimización combinatoria con restricciones y posteriormente se transforma en una formulación QUBO, apta para su mapeo a un Hamiltoniano de Ising.

El caso de uso teórico planteado alcanza 444 variables binarias, una escala no abordable mediante simulación clásica de estado completo con los recursos disponibles. Por ello, la validación experimental se realiza sobre una instancia reducida de 15 variables, diseñada para conservar los elementos esenciales de la formulación y permitir una comparación rigurosa frente a una referencia clásica exacta. La implementación se desarrolla en Qiskit, utilizando QAOA, primitivas de estimación y muestreo, un optimizador clásico COBYLA y un proceso posterior de filtrado de soluciones factibles.

Los resultados muestran que el flujo QUBO-QAOA, combinado con muestreo y posprocesado clásico, recupera el óptimo global del caso reducido, validado mediante enumeración exhaustiva. El trabajo no pretende demostrar ventaja cuántica práctica, sino verificar la viabilidad metodológica de formular, mapear e implementar una abstracción temporal del problema de recuperación de aeronaves en un marco híbrido cuántico-clásico. Finalmente, se discuten las limitaciones asociadas a la escala del problema, la simulación clásica, el *hardware* NISQ y la dependencia del posprocesado clásico, así como posibles líneas futuras de extensión hacia modelos más completos.

Palabras clave: recuperación de aeronaves, QUBO, QAOA, computación cuántica, optimización combinatoria, Qiskit.

Abstract

The management of disruptions in airline operations is a highly complex combinatorial problem, particularly when an aircraft failure or unavailability requires flights to be rescheduled within short recovery windows. This thesis studies the application of variational quantum computing techniques to the aircraft recovery problem through the design and validation of a QUBO-QAOA model applied to a simplified version of the problem.

First, the operational context of irregular airline operations is analysed, together with the main classical approaches used for resource recovery, including exact methods, heuristics and time-space network formulations. Based on this framework, an aircraft recovery model is formulated using binary variables for assignment, delay and cancellation decisions, incorporating uniqueness and temporal continuity constraints. The model is first expressed as a constrained combinatorial optimisation problem and is then transformed into a QUBO formulation suitable for mapping onto an Ising Hamiltonian.

The theoretical use case developed in the thesis reaches 444 binary variables, a scale that cannot be addressed through full state-vector classical simulation with the available computational resources. For this reason, the experimental validation is performed on a reduced 15-variable instance, designed to preserve the essential elements of the formulation while enabling a rigorous comparison against an exact classical reference. The implementation is carried out in Qiskit, using QAOA, estimation and sampling primitives, the COBYLA classical optimiser and a classical post-processing stage to filter feasible solutions.

The results show that the QUBO-QAOA workflow, combined with sampling and classical post-processing, recovers the global optimum of the reduced case, as validated through exhaustive enumeration. The purpose of this work is not to demonstrate practical quantum advantage, but rather to verify the methodological feasibility of formulating, mapping and implementing a temporal abstraction of the aircraft recovery problem within a hybrid quantum-classical framework. Finally, the thesis discusses the limitations related to problem scale, classical simulation, NISQ hardware and dependence on classical post-processing, as well as future research directions towards more complete recovery models.

Keywords: aircraft recovery, QUBO, QAOA, quantum computing, combinatorial optimisation, Qiskit.

Índice general

1. Marco operativo de la industria aérea y gestión de disrupciones	1
1.1. Contexto de la planificación y operación de aerolíneas	1
1.2. Naturaleza y clasificación de disrupciones (IROPS)	4
1.2.1. Clasificación por el origen del factor disruptivo	4
1.2.2. Escenarios operativos y taxonomía de Wang	5
1.2.3. Efecto <i>knock-on</i> y propagación sistémica	6
1.3. Ciclo de gestión de disrupciones	7
1.4. Priorización y jerarquía en la recuperación de aeronaves	9
1.4.1. Recuperación de aeronaves como pilar fundamental de la gestión de disrupciones	9
1.4.2. Interconexión y complejidad de las etapas de recuperación	10
1.5. Evolución hacia una gestión con recuperación integrada	11
1.5.1. La limitación del enfoque secuencial por etapas	11
1.5.2. El potencial de la computación cuántica en la integración total	12
2. Modelos clásicos de optimización en la recuperación de recursos	13
2.1. El problema de recuperación como problema de optimización combinatoria	14
2.1.1. La representación de red espacio-temporal	14
2.1.2. Modelos basados en arcos y modelos basados en rutas	15
2.1.3. Opciones de recuperación	17
2.2. Métodos exactos	18
2.3. Métodos heurísticos y metaheurísticos	22

2.4.	Límites del enfoque clásico y motivación del salto cuántico . .	25
3.	Fundamentos de computación cuántica	27
3.1.	Fundamentos físicos: el qubit y sus propiedades	28
3.1.1.	El qubit y la superposición	28
3.1.2.	El entrelazamiento	29
3.1.3.	La medida y su naturaleza probabilística	30
3.1.4.	Puertas y circuitos cuánticos	31
3.2.	Computación cuántica para la optimización: del potencial a la era NISQ	32
3.3.	Formulación de problemas de optimización para el cómputo cuántico	34
3.3.1.	Problemas NP y optimización combinatoria	34
3.3.2.	El modelo QUBO	35
3.3.3.	El modelo de Ising y los Hamiltonianos	36
3.3.4.	El parámetro de penalización	38
3.4.	El algoritmo QAOA	40
3.4.1.	De la evolución adiabática a QAOA	40
3.4.2.	Estructura del circuito QAOA	41
3.4.3.	El bucle de optimización híbrido	42
4.	Modelado y formulación del problema de recuperación de aeronaves	44
4.1.	Formulación matemática	44
4.2.	Definición del caso de uso teórico	50
4.3.	Justificación de la computación cuántica y el modelo de Ising	53
4.4.	Parámetros de penalización	54
4.5.	Reducción del caso de uso para la validación experimental . .	57
5.	Implementación Tecnológica y Desarrollo en Qiskit	62
5.1.	Arquitectura del desarrollo y stack tecnológico	62
5.2.	Del modelo matemático al QuadraticProgram	65
5.2.1.	Parametrización del escenario operativo	65
5.2.2.	Generación dinámica de las variables binarias	66

5.2.3.	Codificación de la función objetivo	67
5.2.4.	Formulación de las restricciones	68
5.2.5.	Estado del modelo antes de la conversión	69
5.3.	Construcción incremental del modelo	70
5.3.1.	Modelo de un único vuelo	70
5.3.2.	Modelo de dos vuelos con conflicto de recurso	71
5.3.3.	Generalización a flota y horario completos	72
5.4.	Conversión a QUBO y mapeo al Hamiltoniano de Ising	74
5.4.1.	De problema con restricciones a formulación QUBO	74
5.4.2.	Mapeo al Hamiltoniano de Ising	75
5.4.3.	Calibración del coeficiente de penalización	76
5.5.	Algoritmo QAOA: construcción del circuito	78
5.5.1.	Materialización del circuito mediante QAOAAnsatz	78
5.5.2.	Elección de la profundidad del ansatz	79
5.5.3.	Transpilación al backend de ejecución	80
5.6.	Bucle híbrido de optimización	82
5.6.1.	Evaluación del valor esperado mediante Estimator	82
5.6.2.	Elección del optimizador clásico: COBYLA	84
5.6.3.	Inicialización de los parámetros variacionales	85
5.6.4.	Ejecución del bucle y registro de la trayectoria	86
5.7.	Extracción y validación de la solución	88
5.7.1.	Muestreo del estado cuántico óptimo	88
5.7.2.	Decodificación del bitstring	89
5.7.3.	Selección de la solución factible de menor coste	90
5.7.4.	Interpretación de la solución final	91
5.8.	Validación clásica del modelo	93
5.8.1.	Justificación del enfoque de fuerza bruta	93
5.8.2.	Diseño del solver y pre-construcción de estructuras	94
5.8.3.	Enumeración y selección del óptimo	95
5.8.4.	Rol del solver en la validación de la prueba de concepto	96
6.	Análisis de resultados, validación y discusión	98
6.1.	Estrategia experimental y dimensionado del caso de uso	98

6.1.1.	Inviabilidad de la simulación a escala objetivo e intermedia	99
6.1.2.	Escenario reducido: 15 variables binarias	99
6.2.	Configuración experimental del caso de 15 variables	102
6.2.1.	Escenario operativo	102
6.2.2.	Estructura del problema QUBO	103
6.2.3.	Parámetros completos del experimento	103
6.3.	Resultados del <i>solver</i> clásico de referencia	106
6.3.1.	Caracterización del espacio de soluciones factibles . . .	106
6.3.2.	Óptimo global del problema	107
6.3.3.	Interpretación operativa del óptimo	107
6.4.	Resultados del algoritmo QAOA	109
6.4.1.	Convergencia del bucle híbrido	109
6.4.2.	Interpretación de la escala energética	111
6.4.3.	Extracción de la solución mediante muestreo	112
6.5.	Comparación cuántica frente a clásica y validación de la prueba de concepto	114
6.5.1.	Equivalencia de las soluciones	114
6.5.2.	Implicaciones de la validación	115
6.5.3.	Alcance y límites de la conclusión	116
6.6.	Limitaciones identificadas	118
6.6.1.	Limitaciones del <i>hardware</i> cuántico actual	119
6.6.2.	Limitaciones de escalabilidad del circuito	120
6.6.3.	La barrera de la simulación clásica	121
6.6.4.	Dependencia de la calidad del muestreo	122
6.6.5.	Síntesis y proyección	123
7.	Conclusiones y líneas de investigación futura	124
7.1.	Conclusiones	124
7.2.	Líneas de investigación futura	127

Capítulo 1. Marco operativo de la industria aérea y gestión de interrupciones

1.1. Contexto de la planificación y operación de aerolíneas

La operación que hay detrás de los grandes aeropuertos del mundo es una de las más complejas que actualmente se lleva a cabo en el mundo moderno. Abarca múltiples dominios (el diseño del horario y la red de rutas, la asignación de flota, la programación y rotación de aeronaves, la planificación de tripulaciones y la gestión de ingresos), todos ellos sujetos a recursos limitados y a una demanda variable. De ellas depende gran parte de la economía mundial, y desde que empezaron a volar los primeros aviones han estado en constante evolución y cambio. La literatura que discute y avanza la investigación en este campo es por este motivo extensa y está constantemente siendo completada con nuevos avances. Para comprender la magnitud de la gestión de interrupciones, es necesario analizar primero el proceso de planificación que las aerolíneas ejecutan de forma ordinaria. Esta planificación se caracteriza por ser un proceso estructurado y secuencial que comienza meses antes del día de la operación, donde cada etapa depende de los resultados de la anterior para garantizar una red de transporte eficiente y rentable. Se dedica mucho tiempo y recursos a esta tarea, lo que hace que cualquier interrupción suponga pérdidas económicas elevadas.

El objetivo de esta planificación es la optimización de recursos limitados para satisfacer una demanda de mercado que no es fija y está sujeta a variaciones. Este ciclo comienza con el diseño del horario, una fase estratégica donde se definen las rutas y las frecuencias de vuelo basándose en análisis de rentabilidad y la disponibilidad física en los aeropuertos. En esta etapa, la aerolínea

debe equilibrar la conectividad de su red con las limitaciones operativas de los nodos aeroportuarios, estableciendo la base sobre la que se construirán todas las operaciones posteriores (Hinsenveld, 2024).

Una vez consolidado el horario, se inicia la asignación de flota. En este paso, se determina qué tipo de aeronave operará cada trayecto, buscando emparejar de la forma más eficiente posible la capacidad de los aviones con la demanda prevista de pasajeros. Una asignación óptima es fundamental para maximizar los ingresos, ya que utilizar un avión demasiado pequeño supone una pérdida de ventas potenciales, mientras que uno demasiado grande implica costes operativos innecesarios y asientos vacíos. Es por ello que muchas veces se venden más asientos de los disponibles en la aeronave. La baja de un pasajero y por tanto un asiento vacío es algo que se trata de evitar a toda costa por las pérdidas económicas que esto conlleva.

Posteriormente, se lleva a cabo la programación de vuelos y rotación de aeronaves, donde se asigna una aeronave específica, identificada por su matrícula, a una secuencia de vuelos. Este proceso debe garantizar que cada avión cumpla con los ciclos de mantenimiento técnico obligatorios y que se encuentre en el aeropuerto correcto para su siguiente vuelo programado. Una parte importante de este trabajo será definir un escenario de vuelos entre aeropuertos que permita representar una disrupción de flota de forma controlada. No obstante, como se precisará en los capítulos de modelado e implementación, la validación experimental se centrará en una abstracción temporal simplificada del problema. Por simplicidad se creará un escenario ficticio pero coherente con tiempos de vuelo estándar entre los aeropuertos seleccionados. Finalmente, se aborda la programación de tripulaciones, que consiste en asignar a los pilotos y al personal a los servicios correspondientes. Esta fase está sujeta a normativas legales extremadamente estrictas sobre tiempos de vuelo y descanso, lo que añade una capa adicional de complejidad al sistema. Es importante señalar que, debido a la complejidad y la magnitud de las regulaciones laborales implicadas, la gestión de tripulaciones se considera fuera del alcance del presente trabajo, el cual se centra exclusivamente en la optimización de la flota de aeronaves.

Es crucial destacar que todo este proceso secuencial se diseña bajo unas condiciones ideales y recursos fijos, asumiendo que todos los componentes del sistema funcionarán según lo previsto (Hinsenveld, 2024). Esta rigidez estructural hace que la programación sea altamente vulnerable ante las variaciones inevitables de las operaciones reales, donde factores como el clima, averías mecánicas o retrasos en la gestión de tierra pueden invalidar rápidamente la planificación original. Ante estas desviaciones, la aerolínea debe abandonar el plan estático y entrar en una fase dinámica de gestión de interrupciones para intentar restablecer la operatividad de la red. Este trabajo trata principalmente de buscar una manera de optimizar este complejo proceso dinámico de gestión.

1.2. Naturaleza y clasificación de interrupciones (IROPS)

La operación de una aerolínea se diseña bajo un paradigma de condiciones ideales y recursos fijos, donde cada fase de la planificación (horarios, flota, rotaciones y tripulación) depende estrictamente de la anterior para garantizar la rentabilidad (Hinsenveld, 2024). No obstante, en el día de la operación, este equilibrio se ve frecuentemente alterado por las IROPS (*Irregular Operations*). Clausen et al. (2009) definen una interrupción como un estado en el cual la desviación respecto al plan original es de magnitud suficiente como para imponer un cambio sustancial en la programación de los recursos.

La gestión de estas contingencias es crítica para la industria, no solo por la complejidad logística, sino por su severo impacto financiero. Hinsenveld (2024) estima que los costes derivados de las interrupciones suponen entre el 5 y el 10 por ciento de los ingresos anuales de las aerolíneas, lo que representa una pérdida global aproximada de 60.000 millones de dólares anuales. Este impacto económico se desglosa en costes directos de operación (combustible, horas extra), compensaciones a pasajeros y la pérdida de valor de reputación a largo plazo.

1.2.1. Clasificación por el origen del factor disruptivo

La literatura académica divide las fuentes de IROPS según el nivel de control que la aerolínea puede ejercer sobre ellas. Siguiendo a Clausen et al. (2009) y Hinsenveld (2024), se distinguen dos grandes categorías:

- **Factores Internos:** Son fallos originados dentro de la estructura operativa de la propia compañía. Los ejemplos más comunes son las averías mecánicas imprevistas que resultan en estados de *Aircraft On Ground* (AOG), los retrasos en los procesos de mantenimiento programado por falta de piezas o personal, y la indisponibilidad aguda de tripulación por enfermedad o huelgas internas.

- **Factores Externos:** Son eventos ajenos a la gestión de la aerolínea que invalidan la planificación de forma reactiva. La meteorología adversa es el factor predominante, obligando a operar bajo reglas de vuelo instrumental (IFR) y reduciendo la capacidad de las pistas. También se incluyen las restricciones de flujo impuestas por el Control de Tráfico Aéreo (ATC), cierres de aeropuertos por motivos de seguridad o huelgas de servicios de terceros.

1.2.2. Escenarios operativos y taxonomía de Wang

Para facilitar el modelado matemático de la recuperación, Wang et al. (2023) proponen una clasificación basada en tres escenarios operativos, la cual permite al Centro de Control de Operaciones (OCC) aplicar estrategias diferenciadas:

- **Razones de aeronave:** El recurso físico designado no está disponible o no puede operar el vuelo asignado. Es el escenario central de este trabajo, ya que permite la optimización mediante el intercambio de aeronaves o el uso de unidades de reserva para cubrir los vuelos «huérfanos» (Hu et al., 2016).
- **Razones de ruta y aeropuerto:** El canal de comunicación o el nodo de la red están inhabilitados. En estos casos, las opciones de recuperación suelen limitarse a retrasos controlados o cancelaciones, ya que el recurso (avión) está disponible pero el entorno impide su operación.
- **Factores de gestión y pasajeros:** Incluyen interrupciones causadas por fallos en los sistemas de servicios aeroportuarios o incidentes relacionados con los pasajeros (problemas médicos o de seguridad), que afectan marginalmente al tiempo de escala o *Turnaround Time* (TAT).

1.2.3. Efecto *knock-on* y propagación sistémica

La naturaleza más compleja de las IROPS no reside en el evento inicial, sino en su capacidad de propagación a través de la red, fenómeno conocido como efecto *knock-on* (Clausen et al., 2009). Debido a la interconectividad de los recursos, un retraso en la primera parte de una rotación se transfiere automáticamente a los vuelos subsiguientes si el margen de tiempo o *buffer* programado es insuficiente para absorber la desviación.

Maher (2015) señala que esta propagación es la que convierte el problema de recuperación en un desafío NP-difícil. Una decisión tomada de forma aislada para recuperar una aeronave puede generar inviabilidades en las tripulaciones, quienes deben cumplir estrictos tiempos de descanso legal. Por ello, la gestión moderna de disrupciones está evolucionando de enfoques secuenciales y manuales hacia modelos de recuperación integrada que consideran simultáneamente el horario, la flota y las tripulaciones para minimizar el coste sistémico total (Hinsenveld, 2024; Maher, 2015).

1.3. Ciclo de gestión de interrupciones

Una vez que el plan estático diseñado en la fase de planificación estratégica se vuelve inviable debido a una operación irregular, el control de la red pasa de los departamentos de programación al Centro de Control de Operaciones. Este centro es el responsable de ejecutar el ciclo de gestión de interrupciones, cuyo objetivo primordial es devolver el sistema a su estado planificado original con el menor coste y el menor impacto posible en los pasajeros (Clausen et al., 2009).

Este ciclo operativo no es un evento puntual, sino un proceso dinámico y continuo que Maher (2015) y Wang et al. (2023) estructuran en tres fases:

La primera es la detección y evaluación del impacto, que comienza cuando se identifica una desviación significativa respecto al plan (p. ej., una avería técnica o una restricción de tráfico aéreo). En esta etapa, el OCC evalúa no solo el recurso afectado, sino también el efecto *knock-on* sobre el resto de la rotación y las conexiones de pasajeros. Como señala Wang et al. (2023), es común asignar una puntuación de severidad para determinar si la interrupción supera el umbral de tolerancia que obliga a intervenir en el sistema.

Posteriormente, una vez detectada la necesidad de intervenir, los despachadores evalúan diferentes maniobras tácticas. Estas incluyen el retraso controlado de vuelos, el intercambio de aeronaves (*aircraft swaps*), el uso de unidades de reserva o, en última instancia, la cancelación (Hinsenveld, 2024). En la práctica tradicional, este proceso se realiza de forma secuencial: primero se recupera la flota, luego las tripulaciones y finalmente se reubica a los pasajeros (Maher, 2015).

Finalmente, la solución elegida se comunica a los agentes de tierra, tripulaciones y servicios aeroportuarios. Debido a que el entorno es altamente estocástico, el OCC debe monitorizar continuamente la ejecución de la solución para verificar si se ha logrado depurar la red o si la interrupción persiste, requiriendo una nueva iteración del ciclo (Hu et al., 2016).

La eficiencia de este ciclo depende críticamente del tiempo de respuesta, ya que las decisiones a menudo deben tomarse en cuestión de minutos para evitar la pérdida de huecos en pista o el vencimiento de los límites legales de actividad de las tripulaciones (Maher, 2015). Sin embargo, dada la escasez de recursos disponibles durante una crisis, el optimizador se enfrenta a menudo a un conflicto de intereses entre diferentes vuelos. Esta limitación física impone la necesidad de establecer criterios claros de importancia estratégica, teniendo que hacer un análisis de la priorización y jerarquía en la recuperación de aeronaves.

1.4. Priorización y jerarquía en la recuperación de aeronaves

Dada la escasez de recursos y la presión temporal que caracteriza a las operaciones irregulares, el OCC no puede abordar la recuperación de todos los elementos afectados de forma simultánea. En la práctica industrial y académica, la gestión de interrupciones se estructura de forma jerárquica, donde la resolución de las inviabilidades de la flota constituye la base sobre la cual se reconstruye el resto de la operación (Clausen et al., 2009).

1.4.1. Recuperación de aeronaves como pilar fundamental de la gestión de interrupciones

La decisión de centrar este trabajo exclusivamente en la recuperación de la flota de aeronaves responde tanto a una necesidad técnica como a una realidad operativa documentada. Según Eggenberg et al. (2007), el problema de programación integral de una aerolínea es computacionalmente intratable si se intenta resolver como un todo. Por ello, es habitual descomponer el proceso en etapas separadas: aeronaves, tripulación y pasajeros, cada una con sus propios plazos y requisitos de conocimiento específicos.

El presente estudio se enfoca en el denominado *Aircraft Recovery Problem* (ARP) por ser el recurso con mayor rigidez física y logística. Como señalan Clausen et al. (2009), la mayoría de las grandes aerolíneas resuelven las interrupciones de manera secuencial: primero se estabiliza la programación de los aviones, posteriormente se abordan los problemas de las tripulaciones y, finalmente, se gestiona el impacto en los pasajeros. Esta jerarquía es imperativa, ya que sin una asignación física de aeronaves validada (en términos de ubicación, capacidad y mantenimiento), no existe un marco operativo sobre el cual asignar tripulaciones o reubicar pasajeros (Eggenberg et al., 2007).

1.4.2. Interconexión y complejidad de las etapas de recuperación

A pesar de la necesidad de aislar la flota para su modelado, es fundamental reconocer que la recuperación aérea se sustenta sobre tres pilares profundamente interconectados:

- **Aeronaves (Flota):** Representan el recurso físico y técnico. Su recuperación implica gestionar intercambios de matrículas (*tail swaps*) y asegurar el cumplimiento de los ciclos de mantenimiento preventivo, los cuales no pueden ser ignorados bajo ninguna circunstancia operativa (Eggenberg et al., 2007).
- **Tripulación:** Constituye el recurso más sensible a nivel legal. Un retraso en una aeronave puede provocar que la tripulación exceda su límite de horas de actividad, invalidando una solución que técnicamente era factible para el avión (Maher, 2015).
- **Pasajeros:** Son el cliente final y el objetivo de la rentabilidad. Su recuperación busca minimizar el retraso total y evitar la pérdida de conexiones críticas.

La complejidad intrínseca de este proceso reside en que cualquier cambio en una de estas partes genera un efecto inmediato en las otras dos (Maher, 2015). Resolver el problema de forma aislada para la flota, como se propone en este trabajo mediante optimización cuántica, es el primer paso crítico, pero la literatura advierte que este enfoque secuencial puede derivar en soluciones subóptimas para el sistema global (Hu et al., 2016; Maher, 2015). No obstante, la magnitud combinatoria del ARP por sí sola justifica explorar la aplicación de algoritmos de computación cuántica a formulaciones simplificadas de recuperación de flota, como paso previo hacia enfoques más amplios que, en el futuro, puedan incorporar de forma progresiva tripulaciones y pasajeros.

1.5. Evolución hacia una gestión con recuperación integrada

Como se ha mencionado en la sección anterior, actualmente se aborda la recuperación ante una disrupción de manera escalonada y por etapas porque los optimizadores actuales no son capaces de procesar la cantidad de variables que involucra este proceso. Esta fragmentación operativa, comúnmente denominada enfoque de silos, implica que cada departamento (flota, tripulaciones y pasajeros) busca su propia solución óptima de forma aislada, utilizando los resultados de la etapa anterior como restricciones fijas e inamovibles para la siguiente (Maher, 2015).

1.5.1. La limitación del enfoque secuencial por etapas

La literatura reciente subraya que este paradigma secuencial es intrínsecamente ineficiente. Según Clausen et al. (2009), la resolución de las inviabilidades de la flota de manera independiente suele reducir drásticamente el espacio de soluciones para la gestión de tripulaciones. Por ejemplo, un intercambio de aeronaves que minimiza el retraso técnico puede dar lugar a una rotación que infringe los tiempos legales de descanso de los pilotos, invalidando la solución global y forzando cancelaciones de última hora que podrían haberse evitado.

Debido a esto, la tendencia moderna en la investigación operativa se dirige hacia el Problema de Recuperación Integrada (IRP). El objetivo del IRP es consolidar la recuperación del horario, la flota, las tripulaciones y, en modelos más avanzados, el flujo de pasajeros en una única función objetivo sistémica (Maher, 2015). Este enfoque busca alcanzar un óptimo global que minimice simultáneamente los costes operativos y el impacto en la experiencia del cliente, evitando las soluciones subóptimas derivadas de la falta de interacción entre recursos.

1.5.2. El potencial de la computación cuántica en la integración total

A pesar de los beneficios teóricos del modelo integrado, su implementación en tiempo real se enfrenta al desafío de la explosión combinatoria. A medida que se añaden las restricciones de las partes de la aviación, el número de variables binarias crece de tal manera que sobrepasa la capacidad de los algoritmos clásicos.

Es en este punto donde la computación cuántica aparece como una vía de investigación relevante para la gestión futura de IROPS. Gracias a principios como la superposición y el entrelazamiento, ciertos algoritmos cuánticos permiten representar distribuciones sobre espacios de búsqueda de gran tamaño y manipularlas mediante circuitos parametrizados. En este contexto, el uso de modelos QUBO (*Quadratic Unconstrained Binary Optimization*) y su mapeo a Hamiltonianos de Ising permite representar interacciones entre variables binarias en un paisaje energético común, lo que convierte a estos enfoques en candidatos naturales para explorar formulaciones de recuperación progresivamente más integradas. Sobre esta representación operan algoritmos como QAOA (*Quantum Approximate Optimization Algorithm*), que constituye el núcleo metodológico de este trabajo.

El objetivo a largo plazo de esta línea de investigación es avanzar hacia formulaciones más integradas de la recuperación. Si bien este trabajo se centra en estudiar la viabilidad metodológica de QAOA aplicado a una versión simplificada de recuperación de flota, la estructura QUBO planteada proporciona una base conceptual sobre la que podrían incorporarse, en trabajos posteriores, restricciones adicionales relativas a tripulaciones, pasajeros y recursos aeroportuarios. La visión futura no es sustituir de forma inmediata los sistemas actuales, sino explorar si los enfoques cuánticos pueden contribuir, a medida que madure el hardware, a modelos de recuperación más coordinados y sistemáticos.

Capítulo 2. Modelos clásicos de optimización en la recuperación de recursos

El marco operativo expuesto en el Capítulo 1 ha situado la recuperación de flota como la primera fase de la gestión de interrupciones y ha señalado, en la Sección 1.5, las limitaciones del enfoque secuencial por silos frente a una recuperación integrada. Antes de justificar el recurso a la computación cuántica, resulta imprescindible caracterizar con precisión los métodos clásicos con que la investigación operativa ha abordado el *Aircraft Recovery Problem* (ARP) durante las últimas décadas. Este capítulo persigue un doble objetivo: por un lado, mostrar que el problema dispone de un lenguaje de modelado consolidado y de un repertorio maduro de técnicas de resolución; por otro, identificar el punto exacto en el que los métodos encuentran su límite, pues es precisamente esa frontera la que motiva la exploración de alternativas que se desarrolla en los capítulos siguientes. La exposición se organiza de lo general a lo particular: se parte de la naturaleza combinatoria del problema y su representación matemática (Sección 2.1), se revisan a continuación los métodos exactos (Sección 2.2) y los enfoques heurísticos y metaheurísticos (Sección 2.3), y se cierra con un análisis de los límites del enfoque clásico que sirve de puente hacia los fundamentos de la computación cuántica (Sección 2.4).

2.1. El problema de recuperación como problema de optimización combinatoria

El problema de recuperación de aeronaves se define como la tarea de reprogramar las rutas de una flota afectada por una disrupción, con el objetivo de minimizar el coste total mientras se garantiza que, al término de la ventana de recuperación, la operación vuelva a ajustarse al plan previsto y las aeronaves queden posicionadas en los aeropuertos correctos para los vuelos subsiguientes (Su et al., 2021). A diferencia del problema de planificación, que dispone de meses para optimizar el horario, la recuperación opera bajo una restricción temporal severa: la solución debe obtenerse en cuestión de minutos tras el evento disruptivo, lo que confiere a la dimensión computacional del problema una relevancia que no tiene en la fase de planificación (Maher, 2015). Esta presión temporal explica que, durante décadas, la investigación se haya orientado tanto a la formulación de modelos fieles a la complejidad operativa como al desarrollo de técnicas capaces de resolverlos con suficiente rapidez.

Desde el punto de vista de la complejidad computacional, el ARP pertenece a la clase de los problemas NP-difícil. La combinación de variables binarias de asignación, decisiones de cancelación y restricciones de continuidad temporal genera un espacio de soluciones cuyo tamaño crece exponencialmente con el número de vuelos y aeronaves, de modo que la enumeración exhaustiva resulta inviable incluso para instancias de tamaño moderado (Su et al., 2021). Esta caracterización no es un mero tecnicismo: es la raíz de toda la dificultad del problema y la razón última por la que el campo ha invertido tanto esfuerzo en métodos de resolución sofisticados.

2.1.1. La representación de red espacio-temporal

El lenguaje de modelado dominante en la literatura del ARP es la red espacio-temporal (*time-space network*), una estructura que representa la operación

de la flota como un flujo sobre un grafo cuyos nodos codifican pares de estación y tiempo, y cuyos arcos representan las actividades posibles de las aeronaves (Rodrigues et al., 2025; Su et al., 2021). En esta representación coexisten tres tipos de arcos con significados operativos distintos: los arcos de vuelo, que describen un trayecto entre dos aeropuertos con sus horarios de salida y llegada; los arcos de tierra, que representan la permanencia de una aeronave estacionada en un aeropuerto a la espera de su siguiente asignación; y los arcos de vuelo copiados, que modelan las opciones de retraso de un vuelo original desplazando su horario de salida en incrementos discretos predefinidos (Su et al., 2021). Esta última construcción es de particular interés para este trabajo, pues constituye el mecanismo mediante el cual el retraso, una magnitud intrínsecamente continua, se discretiza en un conjunto finito de bandas temporales tratables mediante variables binarias. La formulación del problema desarrollada en el Capítulo 4 adopta precisamente esta lógica: cada vuelo dispone de un conjunto reducido de bandas temporales de inicio, y la decisión de operarlo en una u otra se codifica mediante variables binarias asociadas a las copias correspondientes.

La discretización del tiempo mediante copias de vuelo introduce, no obstante, un compromiso inherente entre precisión y tamaño del problema. Un conjunto amplio de copias permite representar el retraso con granularidad fina, pero multiplica el número de variables y, con él, la dificultad computacional; un conjunto reducido contiene el tamaño del modelo a costa de sobreestimar el retraso real de algunos vuelos (Su et al., 2021). Este equilibrio entre fidelidad temporal y tratabilidad es una de las decisiones de diseño recurrentes en la formulación de modelos de recuperación, y reaparecerá en el dimensionado del caso de uso de este trabajo.

2.1.2. Modelos basados en arcos y modelos basados en rutas

Sobre la base de la red espacio-temporal, la literatura ha desarrollado dos grandes familias de formulaciones que se distinguen por la unidad básica

sobre la que se define la decisión (Su et al., 2021). En los modelos basados en arcos (*arc-based*), las variables de decisión se asocian a arcos individuales de la red, y la ruta de cada aeronave emerge de manera implícita como resultado de las restricciones de balance de flujo que obligan a que el número de aeronaves que entra en cada nodo coincida con el que sale. La función objetivo de estos modelos minimiza la suma de los costes de asignación, de retraso y de cancelación, sujeta a restricciones de cobertura que garantizan que cada vuelo sea operado en exactamente una de sus opciones o bien cancelado, y a restricciones de capacidad que limitan el número de operaciones simultáneas en aeropuertos sometidos a regulación de tráfico (Su et al., 2021). Su principal ventaja es la facilidad de incorporar restricciones de capacidad de aeropuerto; su limitación más relevante es que el flujo agregado no distingue aeronaves individuales, lo que dificulta el tratamiento de interrupciones ligadas a una matrícula concreta, como la avería de un avión específico o sus ciclos de mantenimiento obligatorios.

Los modelos basados en rutas (*path-based*), por el contrario, asocian las variables de decisión a rutas completas, es decir, a secuencias ordenadas de vuelos que una aeronave podría operar a lo largo de la ventana de recuperación (Su et al., 2021). Esta representación permite incorporar de forma implícita, durante la generación de las rutas, restricciones complejas como los tiempos mínimos de rotación o el cumplimiento de los ciclos de mantenimiento, que quedan así eliminadas del modelo central. El precio de esta ventaja es la explosión del número de rutas candidatas, que crece combinatoriamente con el tamaño de la instancia y que vuelve impracticable su enumeración completa, una dificultad que, como se verá, motiva el recurso a las técnicas de generación de columnas. El trabajo seminal de Rosenberger et al. (2003) sobre el reenrutamiento de aeronaves mediante una red de conexiones constituye uno de los ejemplos canónicos de esta familia, y anticipa una estrategia, la de restringir el espacio de búsqueda mediante una preselección heurística, que impregna buena parte de la literatura posterior.

2.1.3. Opciones de recuperación

Con independencia del método elegido, todo modelo de recuperación de flota debe articular un conjunto de acciones operativas mediante las cuales el sistema absorbe el impacto de la disrupción. La literatura ha sistematizado este repertorio en un conjunto de opciones recurrentes (Su et al., 2021). El retraso de vuelos permite reabsorber pequeñas desviaciones aprovechando los márgenes de holgura entre operaciones consecutivas, y se modela, como se ha descrito, mediante copias de vuelo. La cancelación constituye la opción de último recurso, reservada para los casos en que ningún reajuste consigue mantener la operación dentro de los límites admisibles, y lleva asociado el coste más elevado del modelo. La reasignación de aeronaves (*swap*) sustituye la aeronave originalmente prevista por otra disponible en el mismo aeropuerto, redistribuyendo la carga de la disrupción a lo largo de la red. A estas se suman opciones menos frecuentes por su coste, como el empleo de aeronaves de reserva, el traslado de aviones vacíos (*ferrying*), el control de velocidad de crucero para recuperar tiempo en ruta, y, en los modelos que integran la dimensión del pasajero, la reasignación de viajeros a itinerarios alternativos (Su et al., 2021). El modelo desarrollado en este trabajo se concentra en las tres primeras opciones, retraso, cancelación y reasignación de aeronave, por ser las que capturan el núcleo combinatorio de la recuperación de flota sin introducir la complejidad adicional de la gestión de tripulaciones y pasajeros, deliberadamente excluida del alcance según se justificó en el Capítulo 1.

La definición de una disrupción como una desviación respecto al plan de magnitud suficiente para exigir una reprogramación sustancial de los recursos (Clausen et al., 2009), unida a la riqueza de este repertorio de opciones y a la estructura de la red espacio-temporal, configura un problema de optimización combinatoria de gran tamaño cuya resolución eficiente ha dado lugar a dos grandes líneas metodológicas: los métodos exactos, que garantizan la optimalidad de la solución, y los enfoques heurísticos, que renuncian a dicha garantía a cambio de una mayor rapidez. Las secciones siguientes revisan ambas líneas.

2.2. Métodos exactos

La primera gran familia de técnicas aplicadas al problema de recuperación de flota la constituyen los métodos exactos, entendidos como aquellos que garantizan la obtención del óptimo global de la formulación planteada. Dado que, según se ha descrito en el Sección 2.1, el ARP admite una formulación natural como programa lineal entero mixto sobre una red espacio-temporal, el punto de partida de esta familia es la aplicación directa de algoritmos de ramificación y acotación (*branch-and-bound*, Land y Doig (1960)) sobre el modelo entero, habitualmente a través de solucionadores comerciales de propósito general. Este enfoque resuelve la relajación lineal del problema en cada nodo de un árbol de búsqueda y ramifica sobre las variables que toman valores fraccionarios, podando las ramas cuya cota no puede mejorar la mejor solución conocida. Para instancias de tamaño reducido, como la que se emplea en la validación experimental de este trabajo, la enumeración implícita que realiza este procedimiento resulta perfectamente tratable y proporciona una referencia óptima exacta contra la que contrastar otros métodos.

El problema surge cuando se intenta escalar este enfoque directo a las dimensiones de una operación comercial real. El número de variables y restricciones de un modelo basado en arcos crece rápidamente con el número de vuelos, aeronaves y bandas temporales, y el de un modelo basado en rutas se ve afectado por la explosión combinatoria del número de rutas candidatas anticipada en el Sección 2.1. Para una instancia realista, enumerar de antemano todas las rutas posibles que una aeronave podría operar es sencillamente inviable. La respuesta clásica de la investigación operativa a esta dificultad es la generación de columnas, una técnica que evita construir el modelo completo trabajando con un subconjunto reducido de variables, el denominado problema maestro restringido, e incorporando nuevas columnas (rutas) únicamente cuando un subproblema de fijación de precios identifica que pueden mejorar la solución actual. El procedimiento se apoya en la observación de que, en la solución óptima de un problema de gran tamaño, la inmensa mayoría de las variables toma valor nulo, de modo que solo una fracción reducida de las

rutas necesita generarse explícitamente. Cuando esta técnica se integra en un árbol de ramificación y acotación para garantizar la integralidad de la solución, se obtiene el esquema conocido como *branch-and-price*.

La aplicación de la generación de columnas al problema de recuperación de aeronaves fue formalizada, entre otros, por Eggenberg et al. (2007), que resolvieron el ARP mediante un algoritmo de generación de columnas construido sobre una red de bandas temporales (*time-band network*). Esta clase de red reduce el tamaño del modelo agregando las actividades de los aeropuertos en intervalos temporales discretos, lo que disminuye el número de nodos necesarios para describir la red de recuperación y, con ello, acelera la resolución. La idea de acotar deliberadamente el espacio de búsqueda para alcanzar tiempos de respuesta compatibles con la operación reaparece de forma recurrente en la literatura de métodos exactos. Un ejemplo representativo es el trabajo de Rosenberger et al. (2003), que ante la imposibilidad de considerar todas las aeronaves de una flota numerosa, emplea una heurística que preselecciona un subconjunto reducido de aeronaves candidatas al reenrutamiento antes de resolver el modelo exacto sobre ese subconjunto. Esta combinación de una fase heurística de reducción del problema con una fase exacta de resolución es una constante en las implementaciones prácticas, y revela una tensión característica de los métodos exactos aplicados a la recuperación: la garantía de optimalidad teórica debe conciliarse con un límite temporal de minutos, lo que en la práctica obliga a sacrificar parte del espacio de búsqueda.

El refinamiento más relevante de esta línea para el problema de recuperación integrada es la generación de columnas y filas (*column-and-row generation*) desarrollada por Maher (2015). Este método extiende la generación de columnas clásica permitiendo no solo la incorporación dinámica de variables, sino también la de restricciones estructurales, lo que reduce el tamaño del problema maestro y acelera cada resolución de la relajación lineal. Su contribución es doblemente significativa en el contexto de este trabajo. En primer lugar, se aplica al problema de recuperación integrada del horario, la flota y las tripulaciones, es decir, a la versión sistémica del problema que el Capítulo 1 identificó como el horizonte último de la disciplina, y demuestra que es

posible resolverla de forma exacta en tiempos compatibles con la operación. En segundo lugar, Maher (2015) sitúa explícitamente su método frente a la descomposición de Benders, la otra gran técnica de descomposición empleada en recuperación integrada, señalando que esta última no garantiza la optimalidad entera de la solución, mientras que la generación de columnas y filas sí proporciona dicha garantía sin recurrir a aproximaciones. Esta distinción es importante porque delimita con precisión qué ofrece cada método clásico: rapidez frente a garantía de optimalidad, una disyuntiva que estructura buena parte de las decisiones de diseño en el campo.

La vigencia de esta familia de métodos queda patente en la investigación más reciente. Jiang et al. (2025) proponen un método que combina la descomposición de Benders con la generación de columnas para resolver un problema de recuperación integrada que añade la reasignación de puertas de embarque a la recuperación del horario y de la flota. Para contener el coste de generar rutas de aeronave y patrones de asignación de puertas, los autores recurren a un modelo de programación con restricciones que genera estos elementos mediante expresiones lógicas, e introducen técnicas de aceleración específicas para los subproblemas de Benders. Los resultados, obtenidos sobre datos reales, resuelven todas las instancias de prueba por debajo de un umbral de cinco minutos con brechas de optimalidad inferiores al cinco por ciento, y muestran que el enfoque integrado evita las reasignaciones de puerta inviables que provoca la recuperación secuencial. Este trabajo ilustra el grado de sofisticación que han alcanzado los métodos exactos, capaces hoy de abordar problemas integrados de escala industrial, pero también el coste de ingeniería algorítmica que ello exige: descomposiciones a medida, generación dinámica de variables y restricciones, modelos auxiliares de programación con restricciones y técnicas de aceleración diseñadas específicamente para la estructura del problema.

En conjunto, los métodos exactos demuestran que el problema de recuperación, incluso en sus versiones integradas, es resoluble de forma óptima dentro de los plazos operativos cuando se invierte el esfuerzo de modelado y de ingeniería algorítmica necesario. No obstante, esa misma dependencia de

descomposiciones y reducciones ajustadas a la estructura de cada problema, junto con la presión temporal que impone la operación, explica que una parte sustancial de la literatura haya optado por renunciar a la garantía de optimalidad a cambio de la rapidez y la flexibilidad de los métodos heurísticos, que se revisan a continuación.

2.3. Métodos heurísticos y metaheurísticos

Frente a la garantía de optimalidad que ofrecen los métodos exactos revisados en el Sección 2.2, una parte muy considerable de la literatura sobre recuperación de flota ha optado por los métodos heurísticos y metaheurísticos, que renuncian deliberadamente a dicha garantía a cambio de obtener soluciones de buena calidad en tiempos reducidos. Esta elección no responde a una limitación técnica, sino a una realidad operativa: cuando la dimensión del problema o el carácter integrado de la recuperación hacen que el coste computacional de un método exacto resulte incompatible con el límite de minutos que impone la operación, una solución factible y razonablemente buena obtenida con rapidez es preferible a una solución óptima que llega demasiado tarde.

Buena parte de estos enfoques se apoya en metaheurísticas de propósito general, marcos algorítmicos independientes del problema concreto que guían la exploración del espacio de soluciones combinando la búsqueda local con mecanismos para escapar de los óptimos locales. Tres de ellas han tenido una influencia particularmente extensa en la optimización combinatoria y, por extensión, en la recuperación aérea. El recocido simulado, propuesto por Kirkpatrick et al. (1983), se inspira en el proceso físico de enfriamiento controlado de un material. En el contexto de un problema de optimización combinatoria, el algoritmo parte de una solución candidata y explora soluciones vecinas mediante modificaciones locales de dicha solución, por ejemplo cambiar la asignación de un recurso, alterar el orden de una secuencia o activar una opción alternativa. En un problema de recuperación de aeronaves, estos movimientos podrían corresponder a retrasar un vuelo una banda temporal, reasignarlo a otra aeronave o sustituir una operación por una cancelación. La particularidad del recocido simulado es que puede aceptar de forma probabilística algunos de estos cambios aunque empeoren temporalmente el valor de la función objetivo, con una probabilidad que disminuye gradualmente a lo largo de la ejecución. Esta propiedad le permite escapar de óptimos locales en las fases iniciales y converger después hacia una solución de buena calidad.

La búsqueda tabú, formalizada por Glover (1989), dirige la búsqueda local mediante una memoria que prohíbe temporalmente visitar soluciones recientes, evitando así los ciclos y forzando la exploración de nuevas regiones del espacio. Los algoritmos genéticos, cuyo marco conceptual establece Holland (1992), mantienen una población de soluciones candidatas que evoluciona mediante operadores inspirados en la selección natural, como el cruce y la mutación, favoreciendo la combinación de soluciones prometedoras. Estas tres familias, junto con sus numerosas variantes, constituyen el repertorio del que se ha nutrido la mayoría de las aplicaciones heurísticas al ARP.

El traslado de estas metaheurísticas al problema de recuperación ha dado lugar a una literatura abundante. Un ejemplo representativo es el trabajo de Hu et al. (2016), que aborda la recuperación integrada de aeronaves y pasajeros mediante un *Greedy Randomized Adaptive Search Procedure* (GRASP). El método construye iterativamente soluciones combinando una fase de construcción aleatorizada con una de mejora local, y aprovecha las opciones de recuperación descritas en la Sección 2.1, como el intercambio de aeronaves y la reasignación de pasajeros a itinerarios alternativos, para reducir simultáneamente el coste de recuperación y el número de pasajeros afectados. En una línea similar de combinar etapas para abordar la complejidad del problema integrado, Zhang et al. (2015) proponen una heurística en dos fases para la recuperación conjunta de aeronaves y tripulaciones, descomponiendo un problema que en su versión monolítica resultaría difícilmente tratable en subproblemas más manejables resueltos de forma sucesiva. Ambos trabajos ilustran una estrategia recurrente en el campo: emplear la flexibilidad de los métodos heurísticos para abordar versiones integradas del problema que los métodos exactos solo resuelven con un esfuerzo de ingeniería considerable.

Una tendencia más reciente incorpora el análisis de datos operativos históricos al diseño de las heurísticas. El trabajo de Wang et al. (2023) es ilustrativo de esta orientación: a partir de datos de operación de una aerolínea, los autores analizan los patrones de recuperación que los despachadores aplican en la práctica y clasifican las situaciones de interrupción en escenarios diferenciados, para después desarrollar un algoritmo heurístico que imita esas

acciones humanas mediante dos operaciones básicas, el intercambio de matrículas entre vuelos y el reajuste de sus horarios de salida. Este enfoque, que sustituye el ajuste manual de la heurística por la extracción de reglas a partir del comportamiento observado, refleja la creciente disponibilidad de datos operativos y la voluntad de acercar las soluciones automáticas a las prácticas que el personal de los centros de control considera viables.

El balance de esta familia de métodos es, por tanto, el inverso del de los métodos exactos. Su rapidez y su capacidad para acomodar formulaciones complejas e integradas los han convertido en la herramienta dominante en la práctica industrial, pero carecen de garantía de optimalidad y de cualquier cota sobre la calidad de la solución obtenida, su rendimiento depende de un ajuste cuidadoso de los parámetros y resulta sensible a las características de cada instancia. Es interesante observar, además, que algunas de las metaheurísticas más influyentes, como el recocido simulado, se inspiran directamente en fenómenos físicos para guiar la exploración del espacio de soluciones, una idea que la computación cuántica llevará hasta sus últimas consecuencias al codificar el problema en el estado fundamental de un sistema físico real. Este vínculo conceptual entre la optimización inspirada en la física y los métodos cuánticos se retoma en el capítulo siguiente.

2.4. Límites del enfoque clásico y motivación del salto cuántico

La revisión de las dos grandes familias de métodos clásicos permite ahora delimitar con precisión dónde reside la dificultad del problema de recuperación y, con ella, la motivación para explorar paradigmas alternativos. Los métodos exactos descritos en el Sección 2.2 garantizan la optimalidad de la solución, pero la alcanzan a costa de un esfuerzo de modelado y de ingeniería algorítmica considerable, basado en descomposiciones, generación dinámica de variables y restricciones y reducciones del espacio de búsqueda cuidadosamente ajustadas a la estructura de cada problema. Los métodos heurísticos del Sección 2.3 eluden ese coste y ofrecen rapidez y flexibilidad, pero renuncian a la garantía de optimalidad y a cualquier cota sobre la calidad de la solución. Ninguna de las dos familias altera la naturaleza combinatoria del problema, sino que la gestiona mediante estrategias distintas, y ambas operan bajo la misma restricción inflexible: la solución debe estar disponible en cuestión de minutos tras la interrupción.

Conviene subrayar que esta dificultad no procede del problema de recuperación de flota considerado de forma aislada. En su versión aislada, el ARP es perfectamente tratable, como demuestra tanto la práctica industrial como la validación experimental de este trabajo, que resuelve el caso de uso reducido hasta la optimalidad mediante enumeración exhaustiva. El cuello de botella aparece cuando se pretende abordar la recuperación de manera integrada, considerando simultáneamente el horario, la flota, las tripulaciones y los pasajeros. Como se anticipó en el Sección 1.5.1, es la incorporación de las restricciones de las «tres patas» de la operación lo que dispara el número de variables binarias y las interacciones entre ellas, hasta un punto en que la resolución exacta dentro de los plazos operativos se vuelve extraordinariamente exigente. Que incluso los sistemas más recientes basados en solucionadores clásicos necesiten recurrir a una construcción restringida del espacio de búsqueda y a soluciones iniciales de calentamiento para contener el crecimiento combinatorio (Rodrigues et al., 2025) ilustra hasta qué punto la combinatoria

del problema integrado debe domarse de forma activa. Esta es, precisamente, la razón por la que la práctica sigue recurriendo mayoritariamente al enfoque secuencial por silos descrito en el Sección 1.5, pese a que, como advierte la literatura, esa fragmentación tiende a producir soluciones subóptimas para el sistema global (Clausen et al., 2009; Maher, 2015).

El campo se enfrenta, por tanto, a una tensión de fondo que los métodos clásicos gestionan pero no resuelven: la aspiración a una recuperación integrada y de calidad global choca con un coste computacional que crece de forma combinatoria y con un presupuesto temporal muy ajustado. Es en este contexto donde cobra sentido explorar paradigmas de computación que aborden la exploración del espacio de soluciones de un modo cualitativamente distinto al de la búsqueda clásica. La computación cuántica se ha propuesto como una de esas vías, sobre la base de que la codificación del problema en el estado de un sistema cuántico podría permitir explorar simultáneamente regiones del espacio de soluciones que los métodos clásicos recorren de forma esencialmente secuencial. El presente trabajo no pretende demostrar que dicho potencial se traduzca ya en una ventaja práctica, una afirmación que el estado actual de la tecnología no permitiría sostener, sino estudiar la viabilidad y la corrección de aplicar uno de estos algoritmos al problema de recuperación de flota en un escenario controlado. Para abordar ese estudio con rigor es necesario disponer antes de los fundamentos de la computación cuántica y, en particular, del aparato formal que permite traducir un problema de optimización combinatoria como el ARP al lenguaje de un sistema físico. A ese objetivo se dedica el capítulo siguiente.

Capítulo 3. Fundamentos de computación cuántica

El capítulo anterior concluyó que la dificultad del problema de recuperación, en especial en su versión integrada, justifica explorar paradigmas de cómputo que aborden el espacio de soluciones de un modo distinto al de la búsqueda clásica, y advirtió de que acometer ese estudio con rigor exige disponer antes del aparato formal que permite traducir un problema de optimización combinatoria al lenguaje de un sistema físico. Este capítulo proporciona dichos fundamentos. La exposición parte de los conceptos físicos imprescindibles para comprender el resto del trabajo (Sección 3.1), sitúa a continuación la computación cuántica para la optimización en el contexto de la era NISQ (Sección 3.2), desarrolla después el aparato formal que permite expresar un problema de optimización combinatoria en términos aptos para el cómputo cuántico (Sección 3.3) y concluye con la descripción del algoritmo QAOA, sobre el que se sustenta la parte experimental de esta memoria, desarrollada posteriormente en el Capítulo 5

3.1. Fundamentos físicos: el qubit y sus propiedades

La unidad básica de información en la computación clásica es el bit, que toma uno de dos valores posibles, 0 o 1. La computación cuántica generaliza este concepto mediante el bit cuántico o qubit, cuya capacidad para existir en estados sin análogo clásico es la raíz de las propiedades que distinguen este paradigma. Esta sección introduce, sin pretensión de exhaustividad, los conceptos físicos mínimos necesarios para seguir el resto del trabajo: el qubit y la superposición, el entrelazamiento, la medida y la noción de puerta y circuito cuántico. La referencia de fondo para todo este material es el tratado de Nielsen y Chuang (2000).

3.1.1. El qubit y la superposición

A diferencia del bit clásico, un qubit puede encontrarse en una superposición de sus dos estados base, que en notación de Dirac se denotan $|0\rangle$ y $|1\rangle$ y constituyen la denominada base computacional. El estado más general de un qubit se expresa como una combinación lineal de ambos,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad (3.1)$$

donde los coeficientes α y β , denominados amplitudes, son números complejos sujetos a la condición de normalización $|\alpha|^2 + |\beta|^2 = 1$. Mientras un bit clásico solo admite dos configuraciones, el estado de un qubit queda descrito por un punto sobre una esfera, conocida como esfera de Bloch, lo que ilustra la riqueza del espacio de estados accesible (Nielsen & Chuang, 2000).

La verdadera potencia de la superposición se manifiesta al considerar sistemas de varios qubits. Un registro de n qubits no se limita a almacenar una de las 2^n combinaciones posibles de ceros y unos, sino que puede encontrarse en

una superposición de todas ellas de forma simultánea,

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} c_x |x\rangle, \quad \sum_{x \in \{0,1\}^n} |c_x|^2 = 1, \quad (3.2)$$

de modo que el estado vive en un espacio de Hilbert de dimensión 2^n . Esta escala exponencial es la que sustenta la intuición, anticipada al cierre del Sección 2.4, de que un sistema cuántico puede representar y manipular simultáneamente un espacio de soluciones que un computador clásico recorrería de manera esencialmente secuencial.

Conviene, no obstante, matizar esta intuición para evitar una lectura ingenua. Que un registro cuántico represente 2^n amplitudes de forma simultánea no significa que sea posible leerlas todas: como se detalla al tratar la medida, observar el sistema devuelve una única configuración. El aprovechamiento efectivo de la superposición exige orquestar las amplitudes mediante fenómenos de interferencia, de manera que las configuraciones correspondientes a buenas soluciones vean reforzada su probabilidad frente al resto. Precisamente en esa orquestación reside el cometido de algoritmos como QAOA, cuyo desarrollo se presenta en la Sección 5.5.

3.1.2. El entrelazamiento

Una segunda propiedad sin análogo clásico es el entrelazamiento, que describe correlaciones entre qubits de modo que el estado del conjunto no puede expresarse como el producto de los estados individuales de cada qubit. El ejemplo canónico es el de dos qubits en el estado

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle), \quad (3.3)$$

en el que ninguno de los dos qubits posee un estado definido por separado, pero sí lo posee el sistema conjunto: una medida sobre el primer qubit determina de manera instantánea el resultado de medir el segundo. Esta característica, cuyas implicaciones conceptuales fueron objeto de debate desde los prime-

ros años de la teoría cuántica (Einstein et al., 1935), constituye un recurso esencial en computación cuántica, pues permite generar correlaciones entre las variables del problema que amplían el repertorio de estados accesibles más allá de lo que permitiría la mera superposición de qubits independientes (Nielsen & Chuang, 2000).

3.1.3. La medida y su naturaleza probabilística

El acceso a la información contenida en un estado cuántico se produce a través de la medida, un proceso de naturaleza radicalmente distinta a la lectura de un bit clásico. Al medir un qubit en la base computacional, el estado en superposición descrito por la Ecuación (3.1) colapsa a uno de los dos estados base, y el resultado obtenido es intrínsecamente probabilístico: se obtiene el valor 0 con probabilidad $|\alpha|^2$ y el valor 1 con probabilidad $|\beta|^2$,

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2, \quad (3.4)$$

regla conocida como regla de Born (Nielsen & Chuang, 2000). Una vez realizada la medida, la superposición se destruye y las amplitudes originales resultan inaccesibles.

Esta naturaleza probabilística tiene una consecuencia operativa de primer orden para el presente trabajo. La ejecución de un algoritmo cuántico sobre un problema de optimización no devuelve directamente la solución óptima, sino una muestra de configuraciones extraídas conforme a la distribución de probabilidad codificada en el estado final. Obtener una solución útil requiere por tanto repetir la medida un número elevado de veces y procesar el conjunto de resultados, una circunstancia que condiciona la estrategia de extracción de la solución adoptada en este trabajo (Sección 5.7) y que está en el origen de uno de los hallazgos analizados en el capítulo de resultados.

3.1.4. Puertas y circuitos cuánticos

La manipulación del estado de un sistema cuántico se lleva a cabo mediante puertas cuánticas, operaciones que transforman el estado de uno o varios qubits y que matemáticamente se corresponden con transformaciones unitarias, es decir, reversibles y conservadoras de la norma del estado (Nielsen & Chuang, 2000). Un algoritmo cuántico se expresa como un circuito, esto es, una secuencia ordenada de puertas aplicadas sobre un registro de qubits que parte de un estado inicial conocido. Un estado inicial frecuente, y en particular el punto de partida de QAOA, es la superposición uniforme de todas las configuraciones posibles,

$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle, \quad (3.5)$$

que asigna idéntica amplitud a cada una de las 2^n soluciones candidatas y se prepara aplicando una puerta de Hadamard a cada qubit inicializado en el estado $|0\rangle$. De particular relevancia para este trabajo son las puertas parametrizadas, cuyo efecto depende de un parámetro continuo, habitualmente un ángulo de rotación. Estas puertas permiten construir circuitos cuyo comportamiento se ajusta variando dichos parámetros, una propiedad sobre la que se sustenta la familia de algoritmos híbridos a la que pertenece QAOA, desarrollada en la Sección 5.5 y aplicada en la Sección 5.6.

3.2. Computación cuántica para la optimización: del potencial a la era NISQ

El interés por aplicar la computación cuántica a problemas de optimización se asienta sobre la constatación de que ciertos algoritmos cuánticos ofrecen ventajas demostrables frente a sus equivalentes clásicos. El ejemplo más representativo en el ámbito de la búsqueda es el algoritmo de Grover (1996), que permite localizar un elemento marcado dentro de un conjunto no estructurado de N elementos en un número de operaciones del orden de $\sqrt{N}$, frente a las N operaciones que requiere en el peor caso una búsqueda clásica exhaustiva. Esta aceleración cuadrática, modesta en comparación con las aceleraciones exponenciales que se conjeturan para otros problemas, ilustra que la explotación de la superposición y la interferencia descritas en la Sección 3.1 puede traducirse en una reducción tangible del coste computacional, y motiva la pregunta de si beneficios análogos podrían alcanzarse en problemas de optimización combinatoria como el que ocupa a este trabajo.

Existe, sin embargo, una distancia considerable entre la promesa teórica de estos algoritmos y lo que el hardware cuántico actual permite ejecutar. Algoritmos como el de Grover presuponen la disponibilidad de computadores cuánticos tolerantes a fallos, capaces de ejecutar circuitos profundos sobre un gran número de qubits sin que la acumulación de errores degrade el resultado, una tecnología que todavía no existe a la escala necesaria. El estado actual del desarrollo se describe habitualmente mediante el término acuñado por Preskill (2018), era NISQ (*Noisy Intermediate-Scale Quantum*), que caracteriza a los dispositivos disponibles como sistemas de escala intermedia, con un número limitado de qubits y sometidos a un ruido apreciable que, ante la ausencia de corrección de errores, restringe de forma severa la profundidad de los circuitos que pueden ejecutarse de manera fiable.

Esta limitación ha orientado buena parte de la investigación reciente hacia una familia de algoritmos diseñados específicamente para extraer utilidad del hardware NISQ: los algoritmos híbridos variacionales. Su rasgo distintivo es

la combinación de un circuito cuántico parametrizado, del tipo introducido en la Sección 3.1.4, con un optimizador clásico que ajusta de forma iterativa los parámetros de dicho circuito en un bucle de realimentación. De este modo, la parte cuántica se mantiene lo bastante superficial como para resultar ejecutable en presencia de ruido, mientras que la búsqueda de los valores óptimos de los parámetros se delega en un computador clásico. El representante más destacado de esta familia en el ámbito de la optimización combinatoria es el algoritmo cuántico de optimización aproximada (QAOA), propuesto por Farhi et al. (2014), cuyo desarrollo detallado se aborda en la Sección 5.5 y que constituye el núcleo de la parte experimental de esta memoria.

Es importante, llegados a este punto, situar las expectativas con honestidad. La optimización cuántica es un campo de investigación activo en el que conviven oportunidades reales con desafíos abiertos de primer orden, tanto en lo relativo a las limitaciones del hardware como a la propia comprensión teórica del rendimiento de estos algoritmos (Abbas et al., 2024). En el estado actual de la tecnología, la cuestión de si los algoritmos variacionales lograrán una ventaja práctica sostenida sobre los mejores métodos clásicos para problemas de tamaño relevante permanece abierta, y el hardware disponible no permite sostener tal afirmación. Por este motivo, el presente trabajo no persigue demostrar una ventaja cuántica, sino algo más acotado y, a la vez, necesario como paso previo: verificar la viabilidad y la corrección de aplicar QAOA al problema de recuperación de flota en un escenario controlado, contrastando sus resultados frente a un óptimo clásico de referencia. Las limitaciones concretas que el carácter NISQ del hardware impone sobre los resultados obtenidos se analizan en la Sección 6.6.

3.3. Formulación de problemas de optimización para el cómputo cuántico

Los conceptos físicos de la sección anterior describen cómo funciona un computador cuántico, pero no indican por sí solos cómo plantear un problema de optimización en términos que este pueda procesar. Esta sección desarrolla ese puente. Se parte de la caracterización del problema como problema de optimización combinatoria y de su complejidad (Sección 3.3.1), se introduce a continuación el modelo QUBO como formulación de referencia para el cómputo cuántico (Sección 3.3.2), se establece su equivalencia con el modelo de Ising y con un Hamiltoniano cuántico cuyo estado fundamental codifica la solución óptima (Sección 3.3.3), y se cierra con el análisis del parámetro de penalización, la pieza que permite incorporar las restricciones del problema y cuya calibración resultará determinante en la parte experimental (Sección 3.3.4).

3.3.1. Problemas NP y optimización combinatoria

Un problema de optimización combinatoria consiste en encontrar, dentro de un conjunto finito de soluciones posibles, aquella que minimiza o maximiza una determinada función objetivo. En el caso de variables binarias, dicho conjunto está formado por las 2^n combinaciones posibles de n variables, y un problema genérico de esta clase puede expresarse como

$$\min_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \quad \text{sujeto a} \quad g_j(\mathbf{x}) = 0, \quad j = 1, \dots, m, \quad (3.6)$$

donde f es la función objetivo y las funciones g_j codifican las m restricciones que una solución debe satisfacer para ser factible. El problema de recuperación de flota formulado en el Capítulo 4 es un caso particular de esta familia, en el que las variables binarias representan decisiones de asignación y de cancelación, y las restricciones garantizan la coherencia operativa de la solución.

La dificultad de estos problemas no radica en la finitud del espacio de soluciones, sino en su crecimiento exponencial: evaluar de forma exhaustiva las 2^n combinaciones se vuelve impracticable incluso para valores moderados de n . La teoría de la complejidad computacional clasifica los problemas según los recursos necesarios para resolverlos. De forma simplificada, la clase P agrupa los problemas resolubles en tiempo polinómico, mientras que la clase NP comprende aquellos cuya solución, una vez propuesta, puede verificarse en tiempo polinómico. Dentro de este marco, un problema se denomina NP-difícil cuando es al menos tan complejo como los problemas más difíciles de NP. El problema de recuperación de flota pertenece a esta categoría (Maher, 2015), lo que descarta la existencia de un algoritmo clásico eficiente de propósito general y justifica tanto los métodos aproximados revisados en el Capítulo 2 como la exploración de paradigmas alternativos.

La relevancia de esta caracterización para la computación cuántica procede de un resultado fundamental: Lucas (2013) demostró que una amplia variedad de problemas NP admite una formulación natural como la búsqueda del estado de mínima energía de un modelo de Ising. Este resultado es el que permite tender un puente entre los problemas de optimización combinatoria y el hardware cuántico, pues traduce la tarea de minimizar una función objetivo en la de hallar el estado fundamental de un sistema físico. Las dos subsecciones siguientes detallan esta traducción, primero a través del modelo QUBO y después mediante su equivalencia con el modelo de Ising.

3.3.2. El modelo QUBO

El modelo QUBO constituye la formulación de referencia para expresar problemas de optimización combinatoria de manera apta para el cómputo cuántico (Glover et al., 2018). Un problema QUBO consiste en minimizar una función cuadrática de variables binarias sin restricciones,

$$\min_{\mathbf{x} \in \{0,1\}^n} \mathbf{x}^\top Q \mathbf{x}, \quad (3.7)$$

donde Q es una matriz real simétrica que recoge tanto los coeficientes cuadráticos, en sus elementos fuera de la diagonal, como los lineales, que pueden incorporarse a la diagonal aprovechando que para una variable binaria se cumple $x_i^2 = x_i$.

La denominación de problema sin restricciones plantea una aparente dificultad, ya que los problemas de interés práctico, como el de recuperación de flota, están sujetos a restricciones. La técnica que resuelve esta tensión consiste en trasladar las restricciones a la propia función objetivo en forma de penalizaciones. Una restricción de igualdad se incorpora añadiendo a la función objetivo un término cuadrático que se anula cuando la restricción se satisface y toma un valor positivo cuando se viola,

$$\min_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) + P \sum_{j=1}^m (g_j(\mathbf{x}))^2, \quad (3.8)$$

donde $P > 0$ es el parámetro de penalización. Las restricciones de desigualdad, como las de continuidad presentes en el modelo de este trabajo, se reducen previamente a igualdades mediante la introducción de variables de holgura, que se añaden al conjunto de variables binarias del problema. De este modo, cualquier problema de optimización combinatoria con restricciones puede reformularse como un problema QUBO equivalente, en el que la factibilidad deja de imponerse de forma explícita y pasa a estar incentivada por la estructura de la función objetivo. La elección del valor de P no es trivial y se analiza en la Sección 3.3.4.

3.3.3. El modelo de Ising y los Hamiltonianos

El modelo QUBO admite una reformulación equivalente en el lenguaje del modelo de Ising, procedente de la física estadística, que es el que conecta de forma directa con el hardware cuántico. La equivalencia se obtiene mediante un sencillo cambio de variables que transforma cada variable binaria $x_i \in$

$\{0, 1\}$ en una variable de espín $z_i \in \{-1, +1\}$, según la convención habitual

$$x_i = \frac{1 - z_i}{2}. \quad (3.9)$$

Aplicando este cambio a la función objetivo del QUBO se obtiene la energía de un modelo de Ising,

$$H_{\text{Ising}}(\mathbf{z}) = \sum_{i < j} J_{ij} z_i z_j + \sum_i h_i z_i + c, \quad (3.10)$$

donde los coeficientes J_{ij} , denominados acoplamientos, describen la interacción entre pares de espines, los coeficientes h_i representan campos locales sobre cada espín, y c es un término constante, conocido como *offset*, que surge del cambio de variables. Este término desplaza por igual la energía de todas las configuraciones, por lo que no altera cuál de ellas es la óptima, pero debe tenerse en cuenta al interpretar los valores de energía. Los coeficientes concretos que resultan de esta conversión para el caso de uso de este trabajo, incluido el valor del *offset*, se detallan en la Sección 5.4.

El paso final que conecta esta formulación con la computación cuántica consiste en sustituir las variables de espín clásicas por operadores cuánticos. Sustituyendo cada variable z_i por el operador de Pauli Z_i , que actúa sobre el i -ésimo qubit, la energía de Ising se convierte en un operador, denominado Hamiltoniano de coste,

$$H_C = \sum_{i < j} J_{ij} Z_i Z_j + \sum_i h_i Z_i + c \mathbb{I}, \quad (3.11)$$

donde $\mathbb{I}$ es el operador identidad. Un Hamiltoniano describe la energía de un sistema cuántico, y sus autovalores se corresponden con los niveles de energía de los estados propios asociados (Nielsen & Chuang, 2000). El Hamiltoniano de coste de la Ecuación (3.11) es diagonal en la base computacional, de modo que cada estado base es un estado propio cuyo autovalor coincide exactamente con el valor de la función objetivo del QUBO para la configuración correspondiente. En consecuencia, el estado de mínima energía, o estado fundamental, codifica la solución óptima del problema de optimización original

(Lucas, 2013). Con esta construcción, el problema de minimizar una función de coste se ha transformado por completo en el problema físico de encontrar el estado fundamental de un Hamiltoniano, que es precisamente la tarea que aborda el algoritmo QAOA (Sección 5.5).

3.3.4. El parámetro de penalización

La conversión de un problema con restricciones en un problema QUBO, descrita en la Ecuación (3.8), depende de la elección del parámetro de penalización P , una decisión que dista de ser un mero detalle de implementación. El valor de P regula el peso relativo entre dos objetivos en conflicto: minimizar el coste original y respetar las restricciones. Si P es demasiado pequeño, el optimizador puede encontrar ventajoso violar una restricción a cambio de reducir el coste, de modo que el mínimo global del QUBO no se corresponda con una solución factible. Si, por el contrario, P es excesivamente grande, los términos de penalización dominan el paisaje de energía y comprimen las diferencias de coste entre las soluciones factibles, lo que dificulta que el optimizador discrimine entre ellas.

Este segundo riesgo es particularmente relevante en el contexto cuántico. Roch et al. (2023) estudian el efecto de los factores de penalización sobre el espectro de autovalores del Hamiltoniano y muestran que un valor de P inadecuado puede modificar la estructura de dicho espectro de forma que perjudique el rendimiento del algoritmo, al reducir las diferencias de energía relevantes entre los estados de interés. Existe, por tanto, un compromiso que aconseja seleccionar el menor valor de P que garantice la factibilidad de la solución óptima. La literatura ofrece reglas orientativas para esta elección, situando habitualmente el valor adecuado de P en el rango de 1,25 a 2,0 veces los coeficientes de coste del problema (Glover et al., 2018).

No obstante, estas reglas son orientativas y su validez depende de las características del problema concreto. La calibración del parámetro para nuestro caso de uso, así como la discusión del valor finalmente adoptado y de su impacto sobre la distribución de soluciones, se desarrollan en los capítulos de

implementación (Capítulo 5) y resultados (Capítulo 6).

3.4. El algoritmo QAOA

La Sección 3.3 ha reducido el problema de optimización a la tarea de encontrar el estado fundamental de un Hamiltoniano de coste. Esta última sección describe el algoritmo que aborda dicha tarea sobre *hardware* de la era NISQ y que constituye el núcleo metodológico de este trabajo: el algoritmo cuántico de optimización aproximada (QAOA). Se presenta primero su origen conceptual en la computación adiabática (Sección 3.4.1), después su estructura como circuito parametrizado (Sección 3.4.2) y, por último, el bucle de optimización híbrido mediante el que se ajustan sus parámetros (Sección 3.4.3).

3.4.1. De la evolución adiabática a QAOA

El fundamento conceptual de QAOA se encuentra en la computación cuántica adiabática, propuesta por Farhi et al. (2000). La idea de partida es la siguiente: si un sistema cuántico se inicializa en el estado fundamental de un Hamiltoniano sencillo, fácil de preparar, y este se transforma de manera suficientemente lenta hasta convertirse en el Hamiltoniano de coste, el sistema permanecerá en todo momento en su estado fundamental y terminará, por tanto, en el estado fundamental de este último, que codifica la solución del problema. Formalmente, se considera una interpolación entre un Hamiltoniano inicial H_0 y el Hamiltoniano de coste H_C ,

$$H(s) = (1 - s) H_0 + s H_C, \quad s \in [0, 1], \quad (3.12)$$

de manera que en $s = 0$ el sistema se encuentra en el estado fundamental de H_0 y en $s = 1$ en el de H_C . El teorema adiabático garantiza que, si la transformación es suficientemente lenta, el sistema sigue la trayectoria de estados fundamentales sin abandonarla.

La aplicación directa de este principio resulta, sin embargo, problemática sobre *hardware* NISQ, ya que requiere una evolución temporal continua y prolongada que los dispositivos actuales no pueden sostener de forma fia-

ble. QAOA, propuesto por Farhi et al. (2014), surge como una alternativa que sustituye la evolución adiabática continua por una secuencia finita de operaciones discretas, cuyos parámetros se ajustan de forma variacional. De este modo, QAOA conserva la intuición física de guiar el sistema hacia el estado fundamental del Hamiltoniano de coste, pero la materializa mediante un circuito de profundidad finita, ejecutable en el *hardware* disponible, y se inscribe así en la familia de algoritmos híbridos variacionales presentada en la Sección 3.2.

3.4.2. Estructura del circuito QAOA

QAOA opera sobre dos Hamiltonianos. El primero es el Hamiltoniano de coste H_C , definido en la Ecuación (3.11), que codifica la función objetivo del problema y cuyo estado fundamental se desea alcanzar. El segundo es el Hamiltoniano de mezcla,

$$H_M = \sum_{i=1}^n X_i, \quad (3.13)$$

formado por la suma de los operadores de Pauli X_i sobre cada qubit, cuyo papel es inducir transiciones entre los estados de la base computacional y posibilitar así la exploración del espacio de soluciones. El algoritmo se inicializa en la superposición uniforme de la Ecuación (3.5) y aplica a continuación, de forma alternada, las evoluciones asociadas a ambos Hamiltonianos. Tras p repeticiones de este par de operaciones, el estado resultante es

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \dots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^{\otimes n}, \quad (3.14)$$

donde los operadores $e^{-i\gamma_k H_C}$ y $e^{-i\beta_k H_M}$ son puertas parametrizadas del tipo introducido en la Sección 3.1.4, y el conjunto de ángulos $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_p)$ y $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$ constituye los $2p$ parámetros libres del circuito. El número de repeticiones p , denominado profundidad del circuito, es un parámetro fundamental del algoritmo: una mayor profundidad permite, en principio, aproximar con más fidelidad el estado fundamental, al acercarse al límite de la evolución adiabática continua, pero incrementa el número de puertas del

circuito y, con ello, su sensibilidad a los errores sobre *hardware* NISQ (Zhou et al., 2020). La profundidad empleada en este trabajo, junto con el detalle de la construcción del circuito en Qiskit, se presenta en la Sección 5.5.

3.4.3. El bucle de optimización híbrido

Los $2p$ parámetros del circuito no se conocen de antemano, sino que deben determinarse de manera que el estado preparado se aproxime al estado fundamental del Hamiltoniano de coste. Esta determinación se lleva a cabo mediante un bucle de optimización híbrido en el que cooperan un procesador cuántico y uno clásico. En cada iteración, el procesador cuántico prepara el estado $|\gamma, \beta\rangle$ correspondiente a los valores actuales de los parámetros y, mediante medidas repetidas, estima el valor esperado de la energía,

$$F_p(\gamma, \beta) = \langle \gamma, \beta | H_C | \gamma, \beta \rangle, \quad (3.15)$$

que cuantifica la calidad de la solución codificada en el estado: cuanto menor es este valor, más cerca se encuentra el estado de la solución óptima del problema. A continuación, un optimizador clásico utiliza dicho valor para proponer un nuevo conjunto de parámetros que reduzca la energía, y el ciclo se repite hasta converger. La parte cuántica se encarga así de evaluar una función que sería costosa de calcular de forma clásica, mientras que la búsqueda de los parámetros óptimos recae sobre un algoritmo de optimización clásico convencional.

La elección de dicho optimizador clásico no es indiferente, pues su capacidad para navegar el paisaje de la función objetivo, que puede presentar múltiples óptimos locales, condiciona el resultado final, en especial en presencia de los errores característico de los dispositivos NISQ (Pellow-Jarman et al., 2024). El optimizador concreto empleado en este trabajo y los detalles de la configuración del bucle se describen en la Sección 5.6, mientras que las pautas prácticas para la implementación del algoritmo en su conjunto pueden consultarse en la documentación de referencia (IBM Quantum, 2026).

Conviene subrayar, para concluir la sección, una consecuencia de la naturaleza variacional y probabilística del algoritmo, ya anticipada en la Sección 3.1. El resultado de QAOA no es una solución única, sino un estado cuántico que, al medirse de forma repetida, produce una distribución de configuraciones. La solución del problema se obtiene a partir del análisis de esa distribución, un procedimiento cuyos detalles y cuyas implicaciones, centrales para interpretar los resultados de este trabajo, se desarrollan en la Sección 5.7 y en el Capítulo 6.

Capítulo 4. Modelado y formulación del problema de recuperación de aeronaves

4.1. Formulación matemática

Para empezar a modelar el problema debemos partir de la base para cualquier problema de optimización: una función de coste, que en este caso es lo que vamos a tratar de minimizar. Esta función tendrá dos partes principales: la parte propiamente dicha del coste, así como un término que recoja de forma global todas las restricciones que le vamos a imponer a nuestro problema, que definiremos más adelante. En la parte referida al coste, incluiremos tres grupos: el coste de operación de los vuelos, el coste por cancelar un vuelo programado, y el coste por retrasarlo, asignando penalizaciones económicas a cada situación.

Para poder resolver este problema aplicando QAOA, primero debemos plantearlo y modelarlo como un problema QUBO, lo que implica que las restricciones que definamos tienen que estar incluidas directamente en la función de coste, integrándolas mediante penalizaciones cuadráticas. De forma teórica, estas penalizaciones llevan asociado un coeficiente suficientemente grande para que cualquier configuración que viole una restricción resulte energéticamente desfavorable y quede descartada durante la ejecución del algoritmo. Sin embargo, es necesario definir primero las variables a utilizar, la función de coste y las restricciones en forma lineal.

Si consideramos un conjunto de vuelos F y, para cada vuelo $i \in F$, un conjunto de posibles bandas temporales T_i en las que puede ser operado (incluyendo su horario original y posibles retrasos), definimos la variable $x_{it} = 1$ si el vuelo i se opera en la banda t , y $x_{it} = 0$ en caso contrario. Adicionalmente, se introduce una única variable de cancelación z_i por vuelo,

que toma valor 1 si el vuelo no se opera. El retraso no requiere una variable binaria adicional: queda representado directamente por la banda temporal seleccionada, de modo que operar un vuelo en una banda posterior a su salida programada equivale a operarlo con retraso.

Para desarrollar el modelado vamos a partir de un caso sencillo, el cual iremos complicando gradualmente hasta llegar a una definición del problema suficientemente detallada. Empezaremos con una sola aeronave, la cual debe cubrir un número determinado de vuelos. En un momento dado, se le añaden vuelos extra que debe completar. Esto simula que inicialmente había dos naves y que una se avería, dejando a la primera cubriendo todos los vuelos. Consideramos un conjunto total de vuelos F , que resulta de la unión de los vuelos originalmente programados para esta aeronave (F_{orig}) y los vuelos procedentes de la aeronave averiada (F_{extra}), de modo que $F = F_{orig} \cup F_{extra}$. Para cada vuelo $i \in F$, el modelo debe decidir en qué banda temporal t comienza su ejecución dentro de un conjunto de tiempos factibles T_i , o si, por el contrario, el vuelo debe ser cancelado debido a la falta de capacidad.

La función de coste debe minimizar el impacto de esta disrupción, cuantificando el coste por retraso y por cancelación. Denotamos c_{it} al coste asociado a operar un vuelo i en la banda temporal t , x_{it} una variable binaria que vale 1 si un vuelo i es operado en la banda t , C_i al coste fijo de cancelar un vuelo i , y z_i una variable binaria que vale 1 si se cancela un vuelo i :

$$\text{mín} \sum_{i \in F} \left(\sum_{t \in T_i} c_{it} x_{it} + C_i z_i \right) \quad (4.1)$$

En la ecuación anterior, el coste de retraso viene implícito en el término c_{it} : cada banda temporal lleva asociado un coeficiente de coste creciente con el desplazamiento respecto a la salida programada, de modo que operar el vuelo en bandas sucesivas (retrasos de 15, 30, 45 minutos...) resulta progresivamente más caro. La cancelación, en cambio, se modela mediante una única variable por vuelo con un coste fijo C_i . Por tanto, al aumentar el número de bandas temporales definidas crece el número de variables de asignación del modelo,

pero no el de variables de cancelación. Una discretización temporal más fina mejora la precisión del modelo a costa de incrementar el número de variables implicadas; la elección concreta del intervalo de discretización se fija más adelante, en la definición del caso de uso.

A esta función de coste debemos imponer ciertas restricciones. La primera es la de unicidad: para cada vuelo, la suma de todas sus alternativas de operación más su variable de cancelación debe ser exactamente igual a uno. Si una de las variables de operación se activa, la de cancelación debe quedar a cero; si la de cancelación se activa, todas las alternativas de operación deben quedar a cero; y no es admisible que un vuelo quede sin decisión alguna:

$$\sum_{t \in T_i} x_{it} + z_i = 1, \quad \forall i \in F$$

Y otra de continuidad, para que la aeronave no pueda operar más de un vuelo al mismo tiempo. Considerando que un vuelo i operado en un tiempo t tiene una duración d_i y requiere un tiempo de rotación r_i al aterrizar, el intervalo de ocupación del recurso es $[t, t + d_i + r_i]$. Por lo tanto, para cualquier par de variables x_{it} y $x_{jt'}$ cuyos intervalos de ocupación se solapen en el tiempo, se debe cumplir:

$$x_{it} + x_{jt'} \leq 1, \quad \forall i, j \in F, \forall t \in T_i, t' \in T_j$$

Una vez formulado el problema linealmente, hay que transformarlo en un problema QUBO para poder aplicar posteriormente algoritmos QAOA. Para ello debemos agrupar tanto la función de coste como las restricciones en un único Hamiltoniano, con el objetivo de que las restricciones dejen de ser límites rígidos para convertirse en penalizaciones cuadráticas que aumentan el valor de la función si no se cumplen (Glover et al., 2018). El Hamiltoniano tendrá tres elementos:

$$H_{total} = H_{coste} + H_{unic} + H_{cont} \quad (4.2)$$

El Hamiltoniano de coste recoge los costes de operación, retraso y cancelación asociados a cada configuración, y coincide con la función de coste definida en (4.1) sin el operador de minimización: el Hamiltoniano es la función de energía del sistema, y la minimización se realiza posteriormente al buscar su estado fundamental:

$$H_{coste} = \sum_{i \in F} \left(\sum_{t \in T_i} c_{it} x_{it} + C_i z_i \right) \quad (4.3)$$

El Hamiltoniano correspondiente a la restricción de unicidad se construye a partir de su forma lineal siguiendo la transformación estándar para restricciones de igualdad (Glover et al., 2018): se penaliza el cuadrado de la desviación de la suma de decisiones respecto a la unidad, ponderado por un coeficiente de penalización P_{unic} suficientemente grande para asegurar que el sistema priorice cumplir esta igualdad (Roch et al., 2023):

$$H_{unic} = P_{unic} \sum_{i \in F} \left(\sum_{t \in T_i} x_{it} + z_i - 1 \right)^2 \quad (4.4)$$

De este modo, si para un vuelo se activa exactamente una decisión (operarlo en una única banda temporal o cancelarlo), el término correspondiente se anula y no aporta energía. Cualquier otra situación (no tomar ninguna decisión sobre el vuelo, operarlo en dos bandas, u operarlo y cancelarlo simultáneamente) produce una desviación no nula que, al elevarse al cuadrado, añade una penalización estrictamente positiva.

Finalmente, el Hamiltoniano de la restricción de continuidad se modela penalizando únicamente el caso en el que ambas variables en conflicto sean iguales a 1 al mismo tiempo. En lógica binaria, esto se consigue mediante el producto de las dos variables. Si el intervalo de ocupación del vuelo i en la banda t se solapa con el del vuelo j en la banda t' , se añade el siguiente término con

una penalización P_{cont} (Roch et al., 2023):

$$H_{cont} = P_{cont} \sum_{((i,t),(j,t')) \in \Omega} x_{it} \cdot x_{jt'} \quad (4.5)$$

$$\Omega = \{((i,t),(j,t')) \mid i,j \in F, i \neq j, t \in T_i, t' \in T_j, t \leq t' < t + d_i + r_i\} \quad (4.6)$$

Una vez definido en formato QUBO, el siguiente paso es añadir de forma explícita la existencia de más de una nave en el sistema, para pasar de un simple problema de secuenciación de tareas a uno de asignación y recuperación de flota, que es el que realmente queremos estudiar.

Consideramos ahora un conjunto de aeronaves A al que pertenecen todas las aeronaves de nuestra flota, $k \in A$. También debemos incluirlas en las variables binarias que hemos definido hasta ahora. Así, la variable x_{itk} ahora vale 1 si la aeronave k opera el vuelo i en la banda temporal t , y 0 en caso contrario.

Finalmente, los Hamiltonianos que habíamos definido previamente también tienen que ser modificados para incorporar la dimensión de flota:

$$H_{coste} = \sum_{i \in F} \left(\sum_{k \in A} \sum_{t \in T_i} c_{itk} x_{itk} + C_i z_i \right) \quad (4.7)$$

$$H_{unic} = P_{unic} \sum_{i \in F} \left(\sum_{k \in A} \sum_{t \in T_i} x_{itk} + z_i - 1 \right)^2 \quad (4.8)$$

$$H_{cont} = P_{cont} \sum_{k \in A} \sum_{((i,t),(j,t')) \in \Omega} x_{itk} \cdot x_{jt'k} \quad (4.9)$$

Conviene subrayar que, en la restricción de unicidad, el cuadrado abarca la suma completa de alternativas del vuelo sobre todas las aeronaves y bandas temporales: la restricción no compara cada variable de asignación individualmente con la de cancelación, sino que obliga a elegir una única alternativa global por vuelo. Esta formulación es la que posteriormente se implementa en Qiskit mediante una restricción lineal de igualdad por vuelo, en la que se

agregan todas las variables de asignación asociadas a dicho vuelo junto con su variable de cancelación (véase el Capítulo 5).

4.2. Definición del caso de uso teórico

Para evaluar el modelo matemático planteado y dimensionar el reto computacional que motiva la aplicación de algoritmos de computación cuántica, se define un caso de uso cuya magnitud genera una explosión combinatoria representativa de una operación comercial real. El escenario propuesto contempla una red operativa conformada por seis aeropuertos europeos (Madrid, Barcelona, París-CDG, Ámsterdam, Berlín y Roma-FCO), la cual es atendida por una flota de cinco aeronaves, designadas de AC1 a AC5. La ventana temporal de recuperación (*recovery horizon*) se establece en un bloque de seis horas ininterrumpidas, comprendido estrictamente entre las 14:00 y las 20:00 horas.

Dentro de esta franja temporal, el tiempo continuo se ha discretizado en intervalos de 15 minutos para permitir la formulación del problema mediante variables binarias. Por política operativa de la aerolínea virtual modelada, se establece que el retraso máximo permisible para un vuelo antes de forzar su cancelación es de dos horas. Esta combinación de discretización temporal y límite de retraso implica que cada vuelo programado dispone de un conjunto de nueve bandas temporales factibles para iniciar su operación, correspondientes a su hora de salida original y a ocho intervalos sucesivos de retraso. La inclusión de estas variables temporales multiplica el espacio de soluciones del modelo.

El evento disruptivo se introduce en el sistema exactamente a las 14:00 horas. En ese instante, la aeronave AC1 sufre una avería severa en el aeropuerto de Madrid, quedando clasificada como inoperativa para el resto de la jornada. Como consecuencia directa de este fallo técnico, los vuelos que dicha aeronave tenía asignados dentro de la ventana de recuperación quedan desatendidos. El algoritmo de optimización asume entonces la tarea de procesar la reasignación de estos trayectos huérfanos y la reprogramación de los vuelos activos utilizando las cuatro aeronaves restantes de la flota (AC2 a AC5). Para ello, el modelo evalúa la viabilidad temporal de las asignaciones candidatas a través de las variables de asignación, asumiendo que la compatibilidad espacial

ha sido prefiltrada en la definición del escenario. El objetivo es analizar la capacidad de la formulación para absorber el impacto de la disrupción dentro de la abstracción considerada, sin pretender reproducir de forma completa la continuidad espacial de una rotación operativa real.

Para establecer la base del modelo de optimización, la Tabla 4.1 detalla el programa de vuelos original planificado para la flota completa en la franja temporal de estudio. Este horario incluye los tiempos de vuelo calculados entre los nodos de la red y respeta tiempos de rotación realistas en los aeropuertos de destino.

Cuadro 4.1: Programa de vuelos original del caso de uso teórico (14:00 a 20:00)

Aeronave	Vuelo	Origen	Destino	Salida	Llegada	Duración
AC1	101	MAD	BCN	14:00	15:15	1:15
AC1	102	BCN	MAD	16:15	17:30	1:15
AC1	103	MAD	CDG	18:30	20:30	2:00
AC2	201	AMS	BER	14:15	15:30	1:15
AC2	202	BER	BCN	16:30	18:45	2:15
AC3	301	CDG	FCO	14:30	16:30	2:00
AC3	302	FCO	MAD	17:30	20:00	2:30
AC4	401	BCN	CDG	14:00	15:45	1:45
AC4	402	CDG	AMS	16:45	18:00	1:15
AC4	403	AMS	BER	19:00	20:15	1:15
AC5	501	FCO	MAD	14:45	17:15	2:30
AC5	502	MAD	BCN	18:15	19:30	1:15

Una vez definida la situación de partida, se procede a cuantificar el modelo mediante la definición de las variables involucradas. El programa de vuelos de la Tabla 4.1 incluye 12 vuelos que deben ser cubiertos durante el tramo de la disrupción, que comprenden tanto la programación original de las aeronaves operativas como los tres trayectos huérfanos derivados de la avería de AC1. La discretización temporal en tramos de 15 minutos, combinada con la condición de que todo vuelo que no pueda operarse en las dos horas siguientes a su

salida original será cancelado, resulta en 9 bandas temporales posibles para cada trayecto y hace crecer el espacio de soluciones del modelo de manera exponencial.

Al disponer de una flota residual operativa de 4 aeronaves (AC2 a AC5) tras el evento disruptivo, cada vuelo se convierte en una tarea que potencialmente puede ser reasignada a cualquiera de estos recursos. El optimizador no conserva memoria de la asignación original de la aeronave averiada; simplemente intenta encajar los doce trayectos programados en los huecos temporales de las cuatro aeronaves disponibles. Dado que la variable de asignación x_{itk} se define para cada combinación de vuelo i , banda temporal t y aeronave k , el subconjunto de variables de operación asciende a $12 \text{ vuelos} \times 9 \text{ bandas temporales} \times 4 \text{ aeronaves} = 432$ variables binarias. A este valor se le deben sumar las variables de cancelación z_i , que aportan al sistema la flexibilidad de descartar un vuelo si no se encuentra una solución de continuidad válida, añadiendo una variable adicional por cada uno de los 12 vuelos programados. Por tanto, el sistema se compone de un total de 444 variables binarias independientes que conforman el vector de estado del problema.

4.3. Justificación de la computación cuántica y el modelo de Ising

La transición hacia la computación cuántica se fundamenta en la capacidad de mapear problemas de optimización combinatoria a sistemas físicos gobernados por la mecánica cuántica. Según el trabajo de Lucas (2013), es posible encontrar una correspondencia entre cualquier problema NP-completo y el modelo de Ising mediante un número polinómico de pasos. En esta equivalencia, las variables binarias del problema de optimización se transforman en espines de un sistema físico, donde las interacciones entre ellos representan las restricciones de nuestro modelo de recuperación de flota. Esta representación permite tratar la búsqueda de la solución óptima como la búsqueda del estado fundamental (mínima energía) de un Hamiltoniano cuántico, un proceso que aprovecha fenómenos como la superposición y el entrelazamiento para explorar el espacio de soluciones de manera global.

El uso del algoritmo QAOA se justifica como una herramienta diseñada específicamente para abordar problemas de optimización en la era de los dispositivos cuánticos de escala intermedia y con errores (NISQ). Farhi et al. (2014) demuestran que, mediante la aplicación alternada de un Hamiltoniano de coste y un Hamiltoniano de mezcla, el algoritmo puede converger hacia soluciones de alta calidad para problemas combinatorios difíciles. Al aplicar este enfoque al caso de uso teórico de 444 variables, QAOA prepara un estado parametrizado cuya distribución de probabilidad puede concentrarse progresivamente en configuraciones de baja energía, asociadas a soluciones de menor coste dentro de la formulación planteada. Este marco teórico proporciona una base sólida para la implementación en Python y permite plantear el problema de recuperación de aeronaves como un caso adecuado para estudiar la viabilidad de formulaciones QUBO-QAOA aplicadas a problemas de optimización combinatoria, sin asumir ni pretender demostrar una ventaja práctica frente a los métodos clásicos en el estado actual de la tecnología.

4.4. Parámetros de penalización

La sintonización de los parámetros de penalización constituye uno de los desafíos técnicos más críticos en la transición de un modelo de optimización lineal hacia una formulación de tipo QUBO o Hamiltoniano de Ising. La naturaleza de estos problemas reside en que las restricciones, que en la programación clásica actúan como límites rígidos del espacio de búsqueda, se integran directamente en la función objetivo mediante términos cuadráticos ponderados por coeficientes de penalización P . La determinación de estos valores requiere un equilibrio preciso; si las penalizaciones son insuficientes, el algoritmo tiende a favorecer estados de baja energía que violan los límites del problema para minimizar el coste operativo. Por el contrario, si los valores de P son excesivamente elevados, el espectro energético se vuelve abrupto y dominado por las restricciones, lo que oculta las variaciones sutiles de la función de coste original y dificulta la convergencia del algoritmo hacia el óptimo global debido a la reducción del *gap* espectral.

Dentro de la literatura técnica, existen diversas metodologías para abordar esta sintonización. Lucas (2013) establece una cota inferior matemática rigurosa para estos coeficientes, argumentando que, para garantizar que el estado fundamental del sistema corresponda siempre a una solución factible, la penalización P debe ser estrictamente mayor que la ganancia máxima que la función objetivo podría obtener al violar dicha restricción. En el contexto de la recuperación de flota, esto implica que los coeficientes deben ser superiores al coste de cancelar los vuelos implicados, impidiendo que el sistema encuentre rentable operar fuera de los límites definidos. Esta perspectiva es complementada por Glover et al. (2018), quienes proponen un enfoque de *Big M* adaptado a modelos QUBO, donde los parámetros se calibran para superar la magnitud de los coeficientes de la función de coste por un margen de seguridad, asegurando que cualquier infracción sea penalizada con una energía demasiado elevada.

El escalado de estos parámetros encuentra limitaciones físicas en el *hardware* cuántico actual. Roch et al. (2023) demuestran que penalizaciones sobredi-

mencionadas comprimen el espectro de energías de las soluciones válidas, aumentando la probabilidad de que el algoritmo quede atrapado en mínimos locales. Para mitigar este efecto, investigaciones sobre la secuenciación de aeronaves sugieren un enfoque ajustado donde las penalizaciones se sitúan aproximadamente un 10 por ciento por encima de la suma de los costes operativos posibles, buscando mantener el espectro energético lo más suave posible para facilitar la exploración del algoritmo. En contraste, para entornos con ruido característicos de los dispositivos NISQ, se ha propuesto que los parámetros se sitúen en un orden de magnitud superior a los del objetivo para que las señales de las restricciones no se pierdan entre las fluctuaciones de los *qubits*.

La jerarquía de costes en el modelo desarrollado parte de los datos operativos de la industria aérea, donde el coste de cancelación C_i es el factor dominante y debe ser significativamente mayor que cualquier coste de retraso c_{itk} . Una vez definido este coste como el techo de la función objetivo, los coeficientes P se escalan en relación a él. Para algoritmos como QAOA, se establece un rango donde P cumple la cota de factibilidad sin invalidar el análisis de convergencia energética. De este modo, las penalizaciones de unicidad se sitúan entre 1,25 y 2,0 veces el coste de cancelación, y las de continuidad en un rango ligeramente inferior, entre 1,1 y 1,5 veces dicho coste, permitiendo que el optimizador identifique claramente los límites del problema mientras mantiene la capacidad de discernir entre diferentes estrategias de mitigación de retrasos.

La configuración que se detalla a continuación (Tabla 4.2) corresponde al caso de uso teórico definido en la Sección 4.2. Su objetivo es establecer una escala económica coherente para el escenario objetivo de 444 variables, no fijar los valores empleados en la instancia experimental reducida, cuya parametrización final se documenta en los Capítulo 5 y Capítulo 6.

Cuadro 4.2: Configuración de parámetros de coste y penalización del modelo QUBO para el caso de uso teórico

Parámetro	Símbolo	Valor / rango propuesto
Coste de retraso unitario	c_{itk}	Función de t (proporcional al retraso)
Coste de cancelación	C_i	$C_i \approx 10 \times \text{máx}(c_{itk})$
Penalización de unicidad	P_{unic}	$[1,25 \cdot C_i, 2,0 \cdot C_i]$
Penalización de continuidad	P_{cont}	$[1,1 \cdot C_i, 1,5 \cdot C_i]$

Empezando por el coste de retraso, siguiendo la operativa estándar de aerolíneas, se plantea que la operadora pierda 50 € por cada intervalo de 15 minutos de retraso. Denotando por t el índice de la banda temporal, con $t = 0$ para la salida programada:

$$c_{itk} = 50 \cdot t \quad (4.10)$$

El caso teórico contempla nueve bandas temporales por vuelo, correspondientes a la salida original ($t = 0$) y a ocho alternativas retrasadas en incrementos de 15 minutos ($t = 1, \dots, 8$), de modo que el retraso máximo representado equivale a ocho intervalos (dos horas) y el coste máximo de retraso será:

$$\text{máx}(c_{itk}) = 50 \cdot 8 = 400 \text{ €} \quad (4.11)$$

Manteniendo la regla de que la cancelación equivale aproximadamente a diez veces el coste máximo de retraso, el coste de cancelación del caso teórico queda fijado en:

$$C_i = 4000 \text{ €} \quad (4.12)$$

Y las penalizaciones de unicidad y continuidad quedan, aplicando los rangos de la Tabla 4.2, en:

$$P_{unic} \in [5000, 8000] \text{ €} \quad (4.13)$$

$$P_{cont} \in [4400, 6000] \text{ €} \quad (4.14)$$

4.5. Reducción del caso de uso para la validación experimental

El caso de uso definido en la Sección 4.2 establece el escenario operativo de referencia de este trabajo y constituye el objetivo último al que aspira el enfoque desarrollado. Sin embargo, la ejecución del algoritmo QAOA sobre sus 444 variables no es viable en el entorno de simulación clásica empleado: el factor limitante es el coste de simular el circuito cuántico, ya que la representación del estado de un sistema de 444 qubits exige unos recursos de memoria que exceden con varios órdenes de magnitud la capacidad de cualquier estación de trabajo convencional. Esta barrera no es específica de la implementación aquí desarrollada, sino una propiedad fundamental de la simulación clásica de sistemas cuánticos, y es independiente de la dificultad de resolver el problema de forma clásica: lo que impide trabajar a escala objetivo no es la obtención de una referencia exacta, sino la simulación del propio algoritmo cuántico. Por este motivo, la validación experimental se realiza sobre una versión sustancialmente reducida del mismo problema, cuyo dimensionado se define a continuación siguiendo la misma estructura que el caso teórico. La justificación detallada de esta reducción y la exploración de los niveles intermedios de dimensionado se desarrollan en el Capítulo 6.

El escenario reducido contempla una subred operativa conformada por cuatro aeropuertos europeos (Madrid, Barcelona, París-CDG y Ámsterdam), la cual es atendida por una flota de tres aeronaves, designadas de AC1 a AC3. La ventana temporal de recuperación se establece en un bloque de tres horas ininterrumpidas, comprendido estrictamente entre las 14:00 y las 17:00 horas. Al igual que en el caso teórico, el tiempo continuo se discretiza en intervalos de 15 minutos; por política operativa de la aerolínea virtual modelada en este microescenario de validación, el retraso máximo permisible para un vuelo antes de forzar su cancelación se fija en 15 minutos. Esta combinación implica que cada vuelo programado dispone de dos bandas temporales factibles para iniciar su operación, correspondientes a su hora de salida original y a un único intervalo sucesivo de retraso.

El evento disruptivo reproduce el del caso teórico: a las 14:00 horas, la aeronave AC1 sufre una avería severa en el aeropuerto de Madrid y queda clasificada como inoperativa para el resto de la jornada, dejando desatendido el vuelo que tenía asignado. El algoritmo de optimización asume entonces la tarea de procesar la reasignación de este trayecto huérfano y la reprogramación de los vuelos activos utilizando las dos aeronaves restantes de la flota (AC2 y AC3), evaluando la viabilidad temporal de las asignaciones candidatas a través de las variables de asignación, bajo la hipótesis de que la compatibilidad espacial ha sido prefiltrada en el escenario reducido. El objetivo es comprobar el comportamiento de la formulación en una instancia lo bastante pequeña como para ser simulable en los recursos disponibles y, además, contrastable contra un óptimo global exacto.

La Tabla 4.3 detalla el programa de vuelos original del escenario reducido. Este horario incluye los tiempos de vuelo calculados entre los nodos y respeta tiempos de rotación realistas en los aeropuertos de destino.

Cuadro 4.3: Programa de vuelos del caso reducido para validación local

Aeronave	Vuelo	Origen	Destino	Salida	Llegada	Duración
AC1	101	MAD	BCN	14:00	15:15	1:15
AC2	201	BCN	CDG	14:30	16:15	1:45
AC3	301	CDG	AMS	15:00	16:30	1:30

La Figura 4.1 representa gráficamente la situación de partida: tras la avería de AC1, su vuelo queda sin asignar, mientras que las aeronaves operativas mantienen sus trayectos programados dentro de la ventana de recuperación.

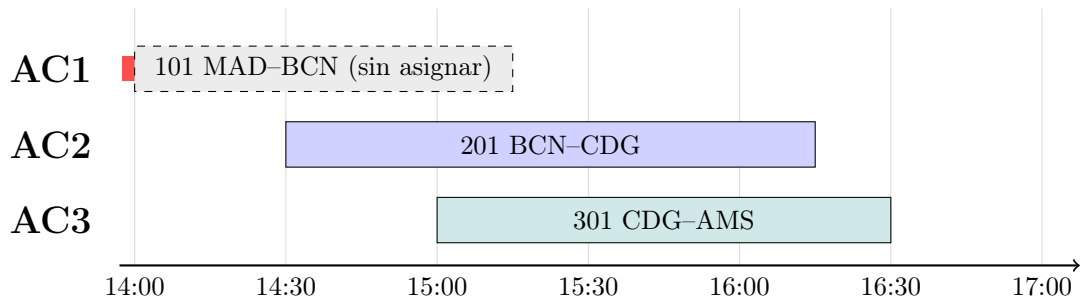


Figura 4.1: Diagrama del problema de recuperación del caso reducido. Tras la avería de la aeronave AC1 a las 14:00, su vuelo 101 queda sin asignar (barra discontinua); las aeronaves AC2 y AC3 operan los vuelos 201 y 301 en sus franjas temporales originales. El optimizador debe decidir la reasignación del vuelo huérfano.

El mismo escenario admite una representación como grafo espacio-temporal (Figura 4.2), habitual en la literatura de recuperación de recursos descrita en el Capítulo 1. En esta representación, cada nodo corresponde a un par (aeropuerto, instante) y cada vuelo es un arco dirigido entre el nodo de salida y el de llegada, de modo que la operación de una aeronave equivale a un camino a través de la red. Esta perspectiva conecta la formulación del problema con su naturaleza combinatoria, ya que cada arco candidato se traduce posteriormente en una variable binaria del modelo.

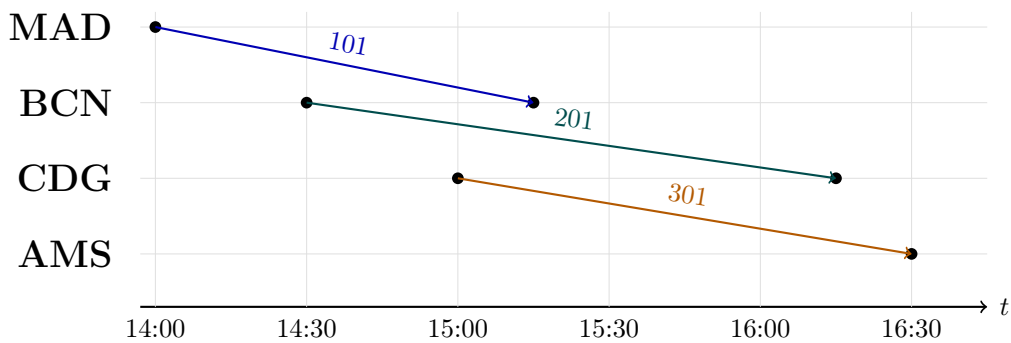


Figura 4.2: Representación del caso reducido como grafo espacio-temporal. Cada nodo corresponde a un par (aeropuerto, instante) y cada arco representa un vuelo, dirigido desde su nodo de salida hasta su nodo de llegada. Una aeronave que opera una secuencia de vuelos describe un camino a través de la red.

Una vez definida la situación de partida, se procede a cuantificar el modelo. El programa incluye 3 vuelos que deben ser cubiertos durante el tramo de la

disrupción (la programación original de AC2 y AC3 más el trayecto huérfano de AC1), y la flota residual operativa se compone de 2 aeronaves, de modo que cada vuelo puede ser reasignado a cualquiera de estos recursos. Dado que la variable de asignación x_{itk} se define para cada combinación de vuelo i , banda temporal t y aeronave k , el subconjunto de variables de operación asciende a $3 \text{ vuelos} \times 2 \text{ bandas temporales} \times 2 \text{ aeronaves} = 12 \text{ variables binarias}$. A este valor se le deben sumar las variables de cancelación z_i , añadiendo una variable adicional por cada uno de los 3 vuelos. Por tanto, el sistema se compone de un total de 15 variables binarias independientes que conforman el vector de estado para la optimización cuántica.

Esta reducción afecta al tamaño de los conjuntos involucrados y conserva los elementos esenciales de la formulación matemática: pluralidad de vuelos y aeronaves, bandas temporales de retraso, restricciones de unicidad, restricciones de continuidad temporal por solapamiento y posibilidad de cancelación. La misma lógica de generación dinámica que produce las quince variables del caso reducido es la que generaría las 444 del caso teórico sin más que ampliar los conjuntos de entrada.

La reducción experimental afecta también a la escala económica empleada en la validación. Mientras que el caso teórico utiliza la parametrización de referencia definida en la Sección 4.4, el caso reducido emplea una escala normalizada, con un coste de retraso de 50 € por banda, un coste de cancelación de 500 € y un coeficiente de penalización QUBO calibrado empíricamente. Esta elección no pretende reproducir la magnitud monetaria absoluta del escenario objetivo, sino conservar la jerarquía relativa entre coste de retraso, coste de cancelación y penalización por violación de restricciones; de hecho, la escala reducida mantiene la misma regla de proporcionalidad del caso teórico, ya que al existir una única banda de retraso el coste máximo de retraso es de 50 € y la cancelación se fija en diez veces dicho valor. La justificación de la calibración empírica se desarrolla en el Capítulo 5.

La caracterización de la barrera de simulación clásica que motiva esta reducción y la transición desde el escenario objetivo hasta el escenario empíricamente validado se documentan en el Capítulo 6. Baste señalar aquí que el

dimensionado del caso experimental no debe interpretarse como una limitación del modelo, sino como una consecuencia del coste exponencial de simular QAOA en un ordenador clásico, que es lo que impide ejecutar el algoritmo a escala objetivo. Trabajar a esta escala reducida tiene, además, una ventaja metodológica: hace posible comparar el resultado del algoritmo contra un óptimo global garantizado por enumeración exhaustiva, algo que a escala objetivo tampoco sería viable, pero que no es la causa de la reducción sino un beneficio añadido de ella.

Capítulo 5. Implementación Tecnológica y Desarrollo en Qiskit

5.1. Arquitectura del desarrollo y stack tecnológico

La transición del modelo matemático formulado en el Capítulo 4 hacia un entorno computacional ejecutable requiere la integración de un conjunto coordinado de librerías y herramientas que cubran las distintas etapas del *pipeline* de optimización cuántica. La implementación desarrollada en este trabajo se apoya en el ecosistema de Qiskit, el *framework* de computación cuántica de código abierto mantenido por IBM, complementado con librerías estándar de Python para el procesamiento numérico y la optimización clásica. La metodología seguida toma como referencia principal la guía oficial de implementación de QAOA publicada por IBM Quantum (IBM Quantum, 2026), adaptando sus patrones de diseño al problema específico de recuperación de flota.

El tecnológico utilizado se estructura en cuatro capas funcionales claramente diferenciadas. La primera capa, dedicada al modelado del problema, se construye sobre el módulo `qiskit-optimization`, que proporciona la clase `QuadraticProgram` para representar problemas de optimización combinatoria de forma explícita, junto con el conversor `QuadraticProgramToQubo`, encargado de absorber las restricciones lineales en la función objetivo mediante penalizaciones cuadráticas. Esta capa permite mantener una separación limpia entre la formulación matemática y su representación cuántica, facilitando la validación intermedia del modelo.

La segunda capa engloba la construcción y manipulación del circuito cuántico. Sobre el núcleo de Qiskit se utiliza la clase `QAOAAnsatz`, que genera

de forma automática el circuito parametrizado correspondiente al algoritmo QAOA a partir del Hamiltoniano de coste obtenido en la etapa anterior. La preparación del circuito para su ejecución se realiza mediante el sistema de transpilación de Qiskit a través de `generate_preset_pass_manager`, que adapta las puertas lógicas del circuito abstracto a la topología y al conjunto de puertas nativas del *backend* objetivo.

La tercera capa corresponde a la ejecución cuántica propiamente dicha, delegada a las primitivas de Qiskit Runtime. Concretamente, se emplean dos primitivas complementarias: **EstimatorV2**, encargada de evaluar el valor esperado del Hamiltoniano $\langle H \rangle$ durante el bucle variacional de optimización, y **SamplerV2**, utilizada en la fase final de muestreo para extraer la distribución de probabilidad sobre los estados de la base computacional.

Estas primitivas pueden dirigirse a dos tipos de destino. El primero es un procesador cuántico físico real, en el que el circuito se ejecuta sobre qubits materiales y el resultado está sujeto al ruido, la decoherencia y los tiempos de cola propios de la infraestructura disponible en la nube. El segundo es un simulador, un programa que se ejecuta íntegramente en un ordenador clásico convencional y que reproduce numéricamente el comportamiento que tendría un procesador cuántico ideal, calculando de forma determinista la evolución del estado cuántico sin recurrir a ningún dispositivo físico. La diferencia es esencial: el procesador real es *hardware* cuántico, mientras que el simulador imita ese *hardware* mediante cálculo clásico.

En este trabajo se utiliza la segunda vía: como entorno de ejecución se emplea el simulador local **AerSimulator** del módulo `qiskit-aer`, que reproduce el comportamiento de un procesador cuántico ideal, sin ruido. Esta elección permite validar la corrección del algoritmo en un entorno determinista y reproducible, libre de los efectos de ruido del *hardware* actual y sin incurrir en los tiempos de cola asociados a la infraestructura cuántica en la nube. La contrapartida es que el coste de esta simulación clásica crece exponencialmente con el número de qubits, una limitación que condiciona el tamaño del problema abordable y que se analiza en detalle en el Capítulo 6.

Finalmente, la cuarta capa gestiona el bucle clásico de optimización de los parámetros variacionales γ y β que definen el circuito QAOA. Para ello se recurre a la función `minimize` de `scipy.optimize`, empleando el método COBYLA (Constrained Optimization BY Linear Approximation), un algoritmo de optimización sin gradientes especialmente adecuado para superficies de coste obtenidas a partir de mediciones cuánticas, como se justificará en detalle en la sección 5.6. La interacción entre estas cuatro capas configura un flujo de trabajo iterativo característico de los algoritmos variacionales híbridos. El optimizador clásico propone un vector de parámetros, el circuito QAOA se ejecuta con dichos parámetros sobre el simulador, la primitiva `Estimator` devuelve el valor esperado del Hamiltoniano, y el optimizador actualiza los parámetros buscando minimizar dicho valor. Este ciclo se repite hasta alcanzar un criterio de convergencia, momento en el que se realiza un muestreo masivo del estado cuántico resultante para extraer la solución combinatoria óptima. Las siguientes secciones detallan cada uno de estos componentes en el orden lógico en que se incorporan al desarrollo.

5.2. Del modelo matemático al QuadraticProgram

La materialización del modelo matemático formulado en el Capítulo 4 se realiza mediante la clase `QuadraticProgram` del módulo `qiskit-optimization`, la cual permite declarar de forma estructurada las variables binarias, la función objetivo y las restricciones lineales que conforman el problema de recuperación de flota. Esta representación intermedia resulta especialmente conveniente porque mantiene la formulación en su forma lineal original, facilitando tanto la verificación del modelo como su posterior conversión automática a un problema QUBO mediante el conversor adecuado, como se detallará en la Sección 5.4.

La construcción del programa cuadrático se ha organizado en cuatro bloques funcionales que reflejan directamente la estructura del modelo matemático: la parametrización del escenario operativo, la generación de las variables binarias de decisión, la codificación de la función objetivo y la formulación de las restricciones de unicidad y continuidad.

5.2.1. Parametrización del escenario operativo

El primer paso consiste en encapsular los datos del caso de uso en estructuras de Python que actúen como única fuente de verdad del modelo. Los vuelos se representan mediante un diccionario donde cada clave identifica un vuelo y su valor recoge la hora de salida programada (expresada en minutos desde medianoche para facilitar las operaciones aritméticas) y la duración de la operación. La flota disponible se modela como una lista de aeronaves, y los parámetros temporales y económicos (número de bandas, intervalo entre bandas, tiempo de rotación y costes unitarios) se definen como constantes a nivel de módulo. Esta separación entre datos y lógica permite modificar el escenario sin alterar el código de modelado, una propiedad esencial para la fase de experimentación.

```

1 vuelos = {
2     '101': {'salida': 840, 'duracion': 75},    # 14:00, MAD-BCN
3     '201': {'salida': 870, 'duracion': 105},   # 14:30, BCN-CDG
4     '301': {'salida': 900, 'duracion': 90},    # 15:00, CDG-AMS
5 }
6 aeronaves = ['AC2', 'AC3']
7 num_bandas = 2
8 intervalo_banda = 15        # minutos
9 tiempo_rotacion = 30        # minutos
10 C_DELAY = 50                # euros por banda de retraso
11 C_CANCEL = 500              # euros por cancelacion

```

Listado 5.1: Parametrización del escenario operativo.

Los valores económicos del listado corresponden exclusivamente a la instancia experimental reducida. En particular, el coste de cancelación de 500 euros representa una escala normalizada para el caso de 15 variables, coherente con la jerarquía definida en el modelo teórico, pero no coincidente con la parametrización absoluta del escenario objetivo de 444 variables (Sección 4.4).

5.2.2. Generación dinámica de las variables binarias

Las variables binarias del modelo se crean de forma programática mediante bucles anidados sobre los conjuntos de vuelos, bandas y aeronaves. Esta aproximación garantiza que el código sea independiente del tamaño del escenario: añadir un vuelo, una aeronave o una banda temporal no requiere modificar la lógica de generación, sino únicamente actualizar las estructuras de datos descritas en el bloque anterior. Para cada vuelo se declara primero su variable de cancelación z_i , y a continuación se generan las variables de asignación x_{itk} para todas las combinaciones posibles de banda temporal y aeronave.

```

1 mod = QuadraticProgram('Recuperacion_Flota_v2')
2 for i in vuelos:
3     mod.binary_var(name=f"z_{i}")
4 for i in vuelos:
5     for t in range(num_bandas):
6         for k in aeronaves:
7             mod.binary_var(name=f"x_{i}_t{t}_{k}")

```

Listado 5.2: Generación de variables binarias.

La nomenclatura adoptada para los identificadores de variable (z_i y $x_{i_t_k}$) reproduce literalmente los subíndices de la formulación matemática, lo que facilita la trazabilidad entre el modelo teórico y su implementación y permite interpretar directamente las soluciones devueltas por el algoritmo.

5.2.3. Codificación de la función objetivo

La función objetivo definida en la Ecuación (4.3) se traduce a código mediante un diccionario que asocia cada nombre de variable con su coeficiente lineal. Las variables de cancelación reciben como coeficiente el coste fijo C_i , mientras que las variables de asignación reciben un coeficiente proporcional al índice de la banda temporal, materializando así el coste lineal de retraso $c_{itk} = t \cdot C_DELAY$. Una vez construido el diccionario completo, se invoca el método `minimize` para fijar el sentido del problema.

```
1 costes = {}
2 for i in vuelos:
3     costes[f"z_{i}"] = C_CANCEL
4     for t in range(num_bandas):
5         coste_banda = t * C_DELAY
6         for k in aeronaves:
7             costes[f"x_{i}_t_{t}_{k}"] = coste_banda
8 mod.minimize(linear=costes)
```

Listado 5.3: Codificación de la función objetivo lineal.

Es importante destacar que en esta etapa la función objetivo contiene exclusivamente los términos de coste operativo. Los términos cuadráticos asociados a las penalizaciones por incumplimiento de restricciones no se introducen manualmente, sino que serán generados automáticamente por el conversor a QUBO a partir de las restricciones lineales que se definen a continuación.

5.2.4. Formulación de las restricciones

Las restricciones del modelo se declaran en su forma lineal nativa mediante el método `linear_constraint`, que recibe un diccionario de coeficientes, un sentido (`==`, `<=` o `>=`), un término independiente y un nombre identificativo. La restricción de unicidad, que exige que cada vuelo sea operado en exactamente una combinación de banda y aeronave o, en su defecto, cancelado, se construye recorriendo el conjunto de vuelos y agregando para cada uno de ellos la suma de sus variables asociadas con coeficiente unitario:

```
1 for i in vuelos:
2     vars_vuelo = {f"x_{i}_t{t}_{k}": 1
3                   for t in range(num_bandas)
4                   for k in aeronaves}
5     vars_vuelo[f"z_{i}"] = 1
6     mod.linear_constraint(
7         linear=vars_vuelo, sense='==', rhs=1,
8         name=f"unic_{i}"
9     )
```

Listado 5.4: Restricción de unicidad.

La restricción de continuidad, que impide que una misma aeronave opere dos vuelos cuyos intervalos de ocupación se solapen, requiere un razonamiento previo para identificar qué pares de variables son incompatibles. Para cada combinación ordenada de dos vuelos (i, j) y cada par de bandas temporales (t_i, t_j) , se calculan los intervalos de ocupación $[\text{inicio}, \text{fin}]$ considerando la hora de salida ajustada por la banda, la duración del vuelo y el tiempo de rotación posterior. Si dichos intervalos se solapan según el criterio $\text{inicio}_i < \text{fin}_j \wedge \text{inicio}_j < \text{fin}_i$, se añade una restricción lineal que limita la suma de ambas variables a uno, prohibiendo así que ambas tomen valor verdadero simultáneamente:

```
1 vuelos_lista = list(vuelos.keys())
2 for k in aeronaves:
3     for idx1, i in enumerate(vuelos_lista):
4         for idx2, j in enumerate(vuelos_lista):
5             if idx1 >= idx2:
6                 continue
7             for t_i in range(num_bandas):
```

```

8         for t_j in range(num_bandas):
9             inicio_i = vuelos[i]['salida'] + t_i*
intervalo_banda
10             fin_i = inicio_i + vuelos[i]['duracion'] +
tiempo_rotacion
11             inicio_j = vuelos[j]['salida'] + t_j*
intervalo_banda
12             fin_j = inicio_j + vuelos[j]['duracion'] +
tiempo_rotacion
13             if (inicio_i < fin_j) and (inicio_j < fin_i):
14                 mod.linear_constraint(
15                     linear={f"x_{i}_t{t_i}_{k}": 1,
16                             f"x_{j}_t{t_j}_{k}": 1},
17                     sense='<=', rhs=1,
18                     name=f"cont_{i}_{t_i}_{j}_{t_j}_{k}",
19                 )

```

Listado 5.5: Restricción de continuidad por solapamiento temporal.

La condición $idx1 \geq idx2$ evita la generación de restricciones duplicadas, ya que el solapamiento entre el vuelo i y el vuelo j es simétrico. La generación dinámica de los nombres de las restricciones, que incorporan los identificadores de los vuelos, las bandas y la aeronave implicada, facilita la depuración del modelo y la interpretación de los resultados.

5.2.5. Estado del modelo antes de la conversión

Al finalizar la ejecución de este bloque, el objeto `QuadraticProgram` contiene una representación lineal completa del modelo simplificado de recuperación considerado en este trabajo: las variables de decisión, la función objetivo a minimizar y el conjunto de restricciones que delimitan el espacio factible. Este modelo, sin embargo, no puede aún ser resuelto mediante QAOA, dado que el algoritmo requiere una formulación QUBO en la que las restricciones estén absorbidas en la función objetivo en forma de penalizaciones cuadráticas. Dicha transformación se aborda en la Sección 5.4, una vez consolidada la construcción incremental del modelo que se describe en la siguiente sección.

5.3. Construcción incremental del modelo

La traducción directa del modelo matemático completo descrito en el Capítulo 4 a una implementación funcional en Qiskit presenta un riesgo metodológico evidente: una formulación con decenas de variables y centenares de restricciones, codificada en un único paso, dificulta la detección de errores estructurales y la validación de la lógica subyacente. Por este motivo, el desarrollo del modelo se ha abordado siguiendo una estrategia incremental, partiendo de un caso mínimo verificable y añadiendo progresivamente las dimensiones operativas hasta alcanzar la configuración final empleada en la validación experimental.

Esta aproximación, habitual en la literatura de prototipado de algoritmos cuánticos aplicados a problemas industriales Kaushik et al. (2025), permite aislar la introducción de cada nueva fuente de complejidad y verificar, en cada etapa, que la solución del modelo coincide con el comportamiento operativo esperado. La construcción se ha estructurado en tres hitos sucesivos: un modelo de un único vuelo que valida la lógica básica de decisión, un modelo de dos vuelos con conflicto de recurso que introduce la restricción de continuidad, y la generalización a flota y horario completos mediante bucles parametrizados.

5.3.1. Modelo de un único vuelo

El primer hito del desarrollo corresponde al análisis aislado de un vuelo individual, con el objetivo de verificar que la lógica binaria de decisión retraso-frente-a-cancelación se codifica correctamente en formato `QuadraticProgram` y que el solver es capaz de recuperar la solución óptima trivial. En este escenario se considera un único vuelo con dos bandas temporales factibles (operación puntual u operación retrasada) y la opción adicional de cancelación. El modelo se reduce, por tanto, a tres variables binarias mutuamente excluyentes ligadas por una única restricción de unicidad.

```
1 mod.binary_var("x_21_1430")
```

```

2 mod.binary_var("x_21_1500")
3 mod.binary_var("z_21")
4 mod.minimize(linear={
5     "x_21_1430": 0,
6     "x_21_1500": 30,
7     "z_21": 100
8 })
9 mod.linear_constraint(
10     linear={"x_21_1430": 1, "x_21_1500": 1, "z_21": 1},
11     sense=="=", rhs=1
12 )

```

Listado 5.6: Núcleo del modelo de un único vuelo.

La calibración de los coeficientes de coste responde a una jerarquía operativa coherente: la operación puntual no genera coste (`x_21_1430` con coeficiente cero), un retraso de una banda introduce un coste intermedio (30 unidades) y la cancelación es la alternativa más penalizada (100 unidades). Esta estructura garantiza que el óptimo trivial corresponde a la operación puntual, lo que permite verificar de forma directa que tanto el solver clásico exacto como el algoritmo QAOA recuperan la solución esperada. La validación de este primer hito confirma que la pipeline de modelado, conversión a QUBO y resolución funciona correctamente sobre un caso de complejidad mínima.

5.3.2. Modelo de dos vuelos con conflicto de recurso

El segundo hito introduce la dimensión que justifica el uso de algoritmos de optimización combinatoria: el conflicto por un recurso compartido. En este escenario se modelan dos vuelos que pueden ser cubiertos por una misma aeronave en franjas temporales solapadas, lo que obliga al modelo a decidir entre operar puntualmente uno y retrasar o cancelar el otro. Aparece aquí, por primera vez, la restricción de continuidad en su forma más elemental: una desigualdad lineal que impide que dos variables de asignación incompatibles tomen valor verdadero simultáneamente.

```

1 mod.linear_constraint(
2     linear={"x_102_0945_AC2": 1, "x_202_0945_AC2": 1},
3     sense="<=", rhs=1, name="Conflicto_AC2_BCN"

```

Listado 5.7: Restricción de conflicto en el modelo de dos vuelos.

La introducción de esta restricción es metodológicamente significativa porque obliga al optimizador a explorar combinaciones de decisiones interdependientes. Mientras que en el hito anterior el espacio de soluciones era trivialmente separable (la elección sobre un vuelo no afectaba a ningún otro), aquí la decisión tomada sobre un vuelo condiciona las opciones disponibles para el otro. Este acoplamiento entre variables es precisamente la fuente de complejidad combinatoria que motiva el recurso a técnicas de optimización avanzadas, y su validación a esta escala reducida permite confirmar que la formulación de la restricción es semánticamente correcta antes de generalizarla.

5.3.3. Generalización a flota y horario completos

El tercer hito consolida la lógica validada en los dos pasos anteriores y la generaliza mediante bucles parametrizados, dando lugar a la formulación final descrita en la Sección 5.2. La transición desde los modelos manuales hasta el modelo automatizado requiere que cada componente (variables, función objetivo y restricciones) se genere dinámicamente a partir de las estructuras de datos que describen el escenario operativo. Esta generalización es la que confiere al modelo su escalabilidad: añadir vuelos, aeronaves o bandas temporales no requiere modificar el código de modelado, sino únicamente actualizar los diccionarios y listas de entrada.

La diferencia esencial entre los dos hitos previos y la versión generalizada reside en el tratamiento de la restricción de continuidad. En el modelo de dos vuelos, las restricciones de conflicto se declaraban manualmente, una por una, identificando a mano qué pares de variables eran incompatibles. Esta aproximación es inviable cuando el número de vuelos y bandas crece, ya que el número de pares potencialmente conflictivos escala cuadráticamente. La generalización resuelve este problema mediante la condición geométrica de solapamiento entre intervalos de ocupación, ya descrita en la Sección 5.2,

que permite generar automáticamente todas las restricciones de continuidad necesarias a partir de la información temporal de cada vuelo.

Conviene señalar que el escenario que efectivamente se ejecuta en este trabajo no es la formulación completa de 444 variables binarias planteada en el Capítulo 4 como caso de uso teórico, sino una versión reducida del mismo problema. Las razones que justifican esta reducción y la metodología de dimensionado del caso experimental se detallan en el Capítulo 6, donde se discute la transición desde el escenario objetivo hasta el escenario empíricamente validado. En cualquier caso, la estructura del modelo es independiente de la escala: la misma lógica que genera 15 variables para el caso reducido es la que generaría las 444 variables del caso teórico sin más que ampliar los conjuntos de entrada.

Una vez consolidado el `QuadraticProgram` mediante esta construcción incremental, el modelo se encuentra preparado para la siguiente etapa del flujo: la conversión a una formulación QUBO y su posterior mapeo a un Hamiltoniano de Ising apto para QAOA. Esta transformación se aborda en la Sección 5.4.

5.4. Conversión a QUBO y mapeo al Hamiltoniano de Ising

Una vez construido el `QuadraticProgram` mediante el procedimiento descrito en la Sección 5.3, el modelo se encuentra expresado como un problema de optimización combinatoria con restricciones lineales explícitas. El algoritmo QAOA, sin embargo, opera sobre Hamiltonianos cuánticos cuyo estado fundamental codifica la solución óptima del problema, lo que exige una transformación intermedia: convertir el modelo con restricciones en una formulación QUBO sin restricciones, y mapear dicha formulación a un operador hermítico expresable mediante cadenas de Pauli. Esta sección detalla ambas conversiones y discute la calibración empírica del coeficiente de penalización, una decisión de diseño que condiciona directamente el comportamiento del algoritmo en la fase de optimización.

5.4.1. De problema con restricciones a formulación QUBO

La conversión a QUBO se realiza mediante la clase `QuadraticProgramToQubo` del módulo `qiskit-optimization`, que absorbe las restricciones lineales del modelo dentro de la función objetivo en forma de penalizaciones cuadráticas. El procedimiento subyacente sigue la formulación estándar descrita por Glover et al. (2018): para cada restricción de igualdad $\sum_j a_{ij}x_j = b_i$, se añade a la función objetivo un término $P \cdot (\sum_j a_{ij}x_j - b_i)^2$, y para cada restricción de desigualdad se introducen variables auxiliares de holgura que permiten reformularla como una igualdad equivalente. El coeficiente P regula la magnitud de la penalización aplicada cuando una restricción es violada.

```
1 conversor = QuadraticProgramToQubo(penalty=PENALIZACION_QUBO)
2 qubo = conversor.convert(mod)
```

Listado 5.8: Conversión del modelo a formulación QUBO.

El resultado de esta operación es un nuevo `QuadraticProgram` sin restricciones explícitas en el que la función objetivo contiene ahora términos cuadráticos derivados del desarrollo de las penalizaciones. Este es el formato canónico requerido por la siguiente etapa de la transformación.

5.4.2. Mapeo al Hamiltoniano de Ising

La formulación QUBO obtenida se traduce a un operador cuántico mediante el método `to_ising`, que realiza el cambio de variable estándar $x_i = (1 - z_i)/2$, donde $z_i \in \{-1, +1\}$ representa el autovalor del operador de Pauli Z actuando sobre el qubit i . Tras este cambio de base, el problema QUBO se expresa como un Hamiltoniano de Ising con términos de un cuerpo (campos locales) y términos de dos cuerpos (acoplos), formalmente equivalente a la representación descrita por Lucas (2013):

$$H = \sum_i h_i Z_i + \sum_{i < j} J_{ij} Z_i Z_j + \text{offset} \quad (5.1)$$

```
1 operador_ising, offset = qubo.to_ising()
```

Listado 5.9: Obtención del Hamiltoniano de Ising.

El método devuelve dos objetos: un operador de tipo `SparsePauliOp` que codifica los términos h_i y J_{ij} como combinaciones de cadenas de Pauli, y un escalar de `offset` que recoge la constante aditiva resultante del cambio de variable. Esta constante no afecta a la localización del mínimo del Hamiltoniano (no depende de los z_i), pero debe sumarse al valor esperado devuelto por el algoritmo para recuperar el coste real del problema original. En la configuración utilizada en este trabajo, el `offset` alcanza valores del orden de varias decenas de miles de unidades energéticas, lo que es coherente con la magnitud del coeficiente de penalización elegido.

El operador resultante es el objeto que sirve de *input* al `QAOAAnsatz` en la Sección 5.5 como Hamiltoniano de coste sobre el que se construye el circuito variacional.

5.4.3. Calibración del coeficiente de penalización

La elección del coeficiente de penalización P constituye una de las decisiones de diseño más delicadas del modelo. El marco teórico establecido en la Sección 4.4 fija unos rangos fundamentados en la literatura: tomando como referencia el coste de cancelación C_i como techo de la función objetivo, las penalizaciones de unicidad se sitúan teóricamente entre $1,25 \cdot C_i$ y $2,0 \cdot C_i$, y las de continuidad entre $1,1 \cdot C_i$ y $1,5 \cdot C_i$. Estos rangos proporcionan una referencia inicial para que las violaciones de restricciones queden suficientemente penalizadas frente a los costes de operación (Glover et al., 2018; Lucas, 2013); su adecuación práctica, no obstante, depende de la estructura del espacio factible y del comportamiento del algoritmo variacional. En el desarrollo experimental, sin embargo, la aplicación directa de los valores teóricos no resultó suficiente para que el algoritmo convergiera hacia el óptimo factible. La caracterización empírica del espacio de soluciones del escenario reducido reveló una propiedad particularmente desfavorable: de las $2^{15} = 32,768$ configuraciones binarias posibles, únicamente una pequeña fracción cercana al 0,11 por ciento corresponde a soluciones factibles (la cifra exacta se documenta en el Capítulo 6). Esta densidad extremadamente baja del espacio factible introduce un desafío específico para QAOA: la mayor parte del paisaje energético explorado por el algoritmo corresponde a estados infactibles, y si la separación energética entre el mínimo factible y los estados infactibles vecinos no es suficientemente acusada, el optimizador clásico tiende a converger hacia regiones del espacio dominadas por configuraciones que violan las restricciones. Este fenómeno es consistente con la observación reportada por Roch et al. (2023) sobre la dependencia del coeficiente de penalización óptimo respecto a la densidad de soluciones válidas en el espacio de búsqueda. Los autores demuestran que, en problemas QUBO con espacios factibles esparcidos, los rangos teóricos derivados de la cota de factibilidad de Lucas (2013) pueden resultar insuficientes en presencia de ruido cuántico o de variabilidad estadística asociada al muestreo finito, requiriendo una recalibración empírica por encima del límite teórico.

Sobre esta base, los valores empleados en la implementación final del caso reducido se han calibrado mediante un proceso de ajuste iterativo, fijando $C_{\text{CANCEL}} = 500$ y $P = 3000$. Esta elección sitúa la penalización en seis veces el coste de cancelación de la instancia experimental, un valor sustancialmente superior al rango teórico de referencia de la Sección 4.4, pero coherente con la baja densidad de soluciones factibles observada en el escenario reducido. Esta desviación no debe interpretarse como una inconsistencia respecto al marco teórico, sino como una manifestación práctica del fenómeno documentado por Roch et al. (2023): en escenarios con espacios factibles extremadamente esparcidos, garantizar la dominancia energética del óptimo factible sobre el espectro de estados infactibles requiere coeficientes de penalización mayores que los predichos por el límite teórico de factibilidad estricta.

Esta calibración constituye, por tanto, un resultado metodológico del propio desarrollo: confirma empíricamente que la elección del coeficiente de penalización en formulaciones QUBO destinadas a QAOA depende no solo de la magnitud de la función objetivo, sino también de la estructura combinatoria del espacio factible, y proporciona evidencia adicional a favor del enfoque de calibración adaptativa propuesto en la literatura reciente.

En consecuencia, el trabajo distingue entre una parametrización económica teórica, empleada para dimensionar el caso objetivo de 444 variables (Sección 4.4), y una parametrización experimental normalizada, utilizada para ejecutar y validar el caso reducido de 15 variables. Ambas mantienen la misma jerarquía relativa de costes, pero no comparten la misma escala monetaria absoluta. Con el Hamiltoniano de Ising construido y los parámetros de penalización calibrados, el problema queda completamente especificado en el formato requerido por QAOA. La construcción del circuito variacional propiamente dicho se aborda en la Sección 5.5.

5.5. Algoritmo QAOA: construcción del circuito

Con el Hamiltoniano de Ising definido en la Sección 5.4, el siguiente paso del flujo consiste en construir el circuito cuántico parametrizado que implementa el algoritmo QAOA. Este circuito constituye el componente cuántico del esquema híbrido y será el objeto sobre el que el bucle clásico de optimización ajustará los parámetros variacionales para minimizar la energía esperada del Hamiltoniano. Esta sección detalla la materialización del circuito mediante la clase `QAOAAnsatz` de Qiskit, la elección de la profundidad del ansatz y la transpilación al conjunto de puertas nativas del backend de ejecución.

5.5.1. Materialización del circuito mediante `QAOAAnsatz`

La construcción manual del circuito descrito anteriormente requeriría la descomposición explícita de los operadores exponenciales en puertas cuánticas elementales, una tarea técnicamente compleja y propensa a errores. Qiskit proporciona la clase `QAOAAnsatz`, que automatiza esta construcción a partir del Hamiltoniano de coste y del número de repeticiones, devolviendo un circuito parametrizado listo para su ejecución.

```
1 REPS = 5
2 circuito_qaoa = QAOAAnsatz(cost_operator=operador_ising, reps=REPS)
3 circuito_qaoa.measure_all()
```

Listado 5.10: Construcción del circuito QAOA.

La instrucción `measure_all` añade al final del circuito las operaciones de medición sobre todos los qubits, lo que permitirá posteriormente extraer la distribución de probabilidad sobre los estados de la base computacional en la fase de muestreo. Conviene señalar que esta versión del circuito, con mediciones incluidas, se utiliza únicamente para el muestreo final mediante `Sampler`; para la evaluación del valor esperado mediante `Estimator` en el bucle variacional, se emplea una versión equivalente del circuito sin mediciones, ya que

la primitiva `Estimator` gestiona internamente la obtención de valores esperados sin requerir mediciones explícitas en la base computacional.

5.5.2. Elección de la profundidad del ansatz

Uno de los hiperparámetros más críticos del algoritmo QAOA es el número de repeticiones p , también denominado profundidad del ansatz. Este parámetro controla un compromiso fundamental: profundidades mayores incrementan la capacidad expresiva del circuito y, en teoría, permiten aproximarse más al estado fundamental del Hamiltoniano, pero también aumentan el número de puertas físicas tras la transpilación, lo que degrada la fidelidad de la ejecución en dispositivos cuánticos reales debido a la acumulación de errores de puerta y a los efectos de decoherencia.

La elección de P debe equilibrar, por tanto, expresividad teórica frente a viabilidad práctica. Zhou et al. (2020) estudian sistemáticamente este compromiso y demuestran que, para una amplia clase de problemas combinatorios, la calidad de la solución mejora monótonamente con P hasta alcanzar un régimen de saturación dependiente del problema y del nivel de ruido del hardware. En la misma línea, Sipos et al. (2025) reportan, en el contexto específico de problemas de optimización aeronáutica como el *crew pairing*, que profundidades del orden de $P = 3$ a $P = 8$ ofrecen un equilibrio adecuado entre calidad de la solución y coste computacional en simulaciones de vector de estado.

Sobre esta base, en la implementación final de este trabajo se ha fijado $P = 5$ como valor de compromiso. Esta elección se justifica por tres consideraciones complementarias: en primer lugar, $P = 5$ proporciona expresividad suficiente para resolver el problema reducido manteniendo el coste computacional dentro de límites razonables en el simulador local; en segundo lugar, este valor se encuentra en el rango identificado por Sipos et al. (2025) como óptimo para problemas combinatorios de aviación de tamaño comparable; y en tercer lugar, valores significativamente superiores incrementarían la profundidad del circuito transpilado hasta niveles que comprometerían la ejecución

en hardware real de la familia NISQ, una consideración relevante de cara a la discusión sobre escalabilidad que se aborda en la Sección 6.6.

5.5.3. Transpilación al backend de ejecución

El circuito construido por **QAOAAnsatz** se expresa en términos de un conjunto abstracto de puertas cuánticas que no necesariamente coincide con las puertas nativas implementadas en el procesador físico o el simulador de destino. La adaptación del circuito a las restricciones del backend se realiza mediante un proceso denominado transpilación, que descompone las puertas abstractas en secuencias equivalentes de puertas nativas y, en el caso de hardware real, las enruta de forma compatible con la topología de conectividad del chip.

```
1 backend = AerSimulator()
2 pm = generate_preset_pass_manager(
3     optimization_level=3,
4     backend=backend
5 )
6 circuito_transpilado = pm.run(circuito_qaoa)
```

Listado 5.11: Transpilación del circuito al backend de ejecución.

El gestor de pasos de transpilación generado por **generate_preset_pass_manager** aplica una secuencia predefinida de optimizaciones cuyo nivel de agresividad se controla mediante el parámetro **optimization_level**. El valor 3, máximo disponible, activa todas las optimizaciones de simplificación de circuito, fusión de puertas adyacentes y eliminación de redundancias, produciendo un circuito final con una profundidad significativamente menor que la obtenida con niveles inferiores. Como entorno de ejecución se utiliza **AerSimulator**, el simulador de vector de estado del módulo **qiskit-aer**, que permite la verificación determinista del algoritmo sin incurrir en los tiempos de cola asociados a la infraestructura cuántica en la nube.

La profundidad del circuito transpilado en este entorno alcanza una profundidad de 73 capas, una cifra moderada coherente con un escenario reducido y con el uso de un simulador de vector de estado que no impone restricciones

de conectividad entre qubits. Esta misma operación de transpilación, realizada sobre hardware real con topología de conectividad limitada, produce un incremento sustancial en el número de operaciones del circuito como consecuencia de la inserción de puertas de intercambio (*SWAP gates*) necesarias para enrutar la información entre qubits no adyacentes. El análisis detallado de este fenómeno y sus implicaciones para la escalabilidad del algoritmo se aborda en la Sección 6.6. Con el circuito QAOA construido, parametrizado y transpilado al backend de simulación, el modelo se encuentra preparado para entrar en el bucle de optimización híbrida en el que un optimizador clásico ajustará los vectores $\boldsymbol{\gamma}$ y $\boldsymbol{\beta}$ para minimizar el valor esperado del Hamiltoniano. Esta fase se describe en la Sección 5.6.

5.6. Bucle híbrido de optimización

El circuito QAOA construido en la Sección 5.5 es un circuito parametrizado: su comportamiento depende de los vectores variacionales γ y β cuyos valores óptimos no son conocidos a priori. La determinación de estos parámetros constituye el componente clásico del algoritmo híbrido y se aborda mediante un bucle de optimización iterativa en el que un optimizador clásico propone candidatos de parámetros, el circuito cuántico evalúa el valor esperado del Hamiltoniano para dichos candidatos, y el optimizador actualiza su propuesta sobre la base del resultado. Esta sección describe los tres elementos centrales de este bucle: la primitiva **Estimator** encargada de la evaluación cuántica, la elección del optimizador clásico COBYLA, y la estrategia de inicialización de los parámetros.

5.6.1. Evaluación del valor esperado mediante Estimator

El indicador que el bucle de optimización busca minimizar es el valor esperado del Hamiltoniano de coste sobre el estado preparado por el circuito QAOA:

$$\langle H_C \rangle_{\gamma, \beta} = \langle \gamma, \beta | H_C | \gamma, \beta \rangle \quad (5.2)$$

Esta cantidad escalar resume, para cada combinación concreta de parámetros, la energía promedio del estado cuántico generado por el ansatz, y constituye la función objetivo del problema variacional. En Qiskit, su cálculo se delega a la primitiva **EstimatorV2**, que internamente gestiona la ejecución del circuito y la agregación estadística de los resultados.

```
1 estimator = Estimator(mode=backend)
2 estimator.options.default_shots = 8192
3 estimator.options.simulator = {"seed_simulator": 42}
```

Listado 5.12: Configuración de la primitiva **Estimator**.

El número de *shots*, fijado en 8.192, controla la precisión estadística de la estimación: valores mayores reducen la varianza del estimador a costa de un

mayor tiempo de cómputo. La elección de 8.192 representa un valor intermedio habitual en la literatura, suficiente para que el ruido estadístico no domine la señal pero sin penalizar excesivamente el tiempo de cada iteración del bucle. La semilla del simulador se fija explícitamente para garantizar la reproducibilidad determinista de los resultados. La función objetivo que se entrega al optimizador clásico se define mediante un *wrapper* que encapsula la llamada al `Estimator` y registra el historial de evaluaciones para su análisis posterior:

```

1 historial = []
2 mejor_energia = float('inf')
3 mejores_params = None
4 def objetivo(params):
5     global mejor_energia, mejores_params
6     pub = (circuito_para_estimador_tr, isa_operador, params)
7     res = estimator.run([pub]).result()
8     energia = float(res[0].data.evs)
9     historial.append(energia)
10    if energia < mejor_energia:
11        mejor_energia = energia
12        mejores_params = params.copy()
13    return energia

```

Listado 5.13: Función objetivo del bucle híbrido.

La estructura `pub` (*primitive unified bloc*) agrupa el circuito, el operador a medir y el vector de parámetros en el formato canónico requerido por la API de `EstimatorV2`. La lista `historial` almacena cada valor esperado evaluado durante la optimización, lo que posibilita la construcción del gráfico de convergencia que se analiza en la Sección 6.4. Adicionalmente, el bloque condicional registra el mejor par energía-parámetros encontrado a lo largo de toda la trayectoria, una salvaguarda necesaria dado que algunos optimizadores no garantizan que su iteración final corresponda al mínimo histórico explorado.

5.6.2. Elección del optimizador clásico: COBYLA

La superficie de la función objetivo evaluada mediante mediciones cuánticas presenta dos características que condicionan severamente la elección del optimizador clásico. En primer lugar, no se dispone de información analítica del gradiente, ya que la función se evalúa estadísticamente a partir de un número finito de disparos. En segundo lugar, esta evaluación introduce un nivel intrínseco de ruido estadístico, agravado en hardware real por los errores de puerta y la decoherencia. Los métodos de optimización basados en gradientes son particularmente sensibles a este tipo de superficies, ya que la estimación numérica de derivadas amplifica el ruido y conduce con frecuencia a oscilaciones inestables. Estas limitaciones motivan el uso de optimizadores libres de gradiente, diseñados específicamente para superficies ruidosas. En este trabajo se ha empleado COBYLA (*Constrained Optimization BY Linear Approximation*), un algoritmo de optimización local sin restricciones explícitas que aproxima la función objetivo mediante modelos lineales construidos sobre un simplejo de puntos de evaluación y actualiza dicho simplejo siguiendo una estrategia de región de confianza. Su robustez frente al ruido estadístico y su ausencia de requisitos sobre la diferenciabilidad de la función lo convierten en una elección estándar para algoritmos variacionales cuánticos.

La pertinencia de esta elección está respaldada por la literatura especializada. Pellow-Jarman et al. (2024) realizan un estudio comparativo de optimizadores clásicos aplicados a QAOA y concluyen que COBYLA y SPSA presentan un rendimiento robusto en escenarios de simulación de vector de estado sin ruido, mientras que en presencia de ruido cuántico realista el rendimiento relativo de los distintos optimizadores depende sensiblemente de la profundidad del ansatz y de la estructura del problema. Esta observación es relevante para el alcance de este trabajo: en un entorno de simulación determinista como el utilizado aquí, COBYLA proporciona un comportamiento estable y reproducible; el análisis de su rendimiento en hardware ruidoso quedaría como línea natural de extensión del estudio.

```
1 res_opt = minimize(  
2     objetivo, x0,
```

```

3 method='COBYLA',
4 options={'maxiter': 800, 'rhobeg': 0.3, 'disp': False},
5 )

```

Listado 5.14: Invocación del optimizador COBYLA.

Los hiperparámetros del optimizador se han calibrado experimentalmente. El número máximo de iteraciones (`maxiter=800`) proporciona un presupuesto suficientemente amplio para garantizar la convergencia en el escenario reducido sin incurrir en un coste computacional excesivo. El parámetro `rhobeg`, que controla el tamaño inicial del simplejo de exploración, se fija en 0,3 tras observar que valores mayores generan saltos demasiado agresivos en las primeras iteraciones y valores menores ralentizan excesivamente la exploración inicial del paisaje energético.

5.6.3. Inicialización de los parámetros variacionales

La elección del punto inicial $\mathbf{x}_0 = (\gamma_0, \beta_0)$ tiene un impacto significativo en la calidad de la solución alcanzada, ya que COBYLA, al ser un optimizador local, puede quedar atrapado en mínimos locales si la inicialización es desfavorable. Zhou et al. (2020) estudian sistemáticamente este aspecto y proponen varias estrategias de inicialización, desde la inicialización aleatoria uniforme hasta esquemas más sofisticados basados en la interpolación de soluciones óptimas conocidas para profundidades menores. En este trabajo se ha optado por una inicialización aleatoria estructurada que respeta los rangos físicamente sensatos de cada tipo de parámetro:

- Los parámetros γ_p , asociados al Hamiltoniano de coste, se inicializan uniformemente en el intervalo $[0, \pi]$.
- Los parámetros β_p , asociados al Hamiltoniano de mezcla, se inicializan uniformemente en el intervalo $[0, \pi/2]$.

```

1 np.random.seed(42)
2 x0 = np.zeros(circuito_qaoa.num_parameters)
3 for p in range(REPS):
4     x0[2 * p] = np.pi * np.random.rand() # gamma_p

```

5

```
x0[2 * p + 1] = (np.pi / 2) * np.random.rand() # beta_p
```

Listado 5.15: Inicialización de los parámetros variacionales.

Estos rangos se justifican por la estructura periódica del ansatz: el Hamiltoniano de mezcla $H_M = \sum_i X_i$ tiene autovalores enteros, lo que confiere al operador $e^{-i\beta H_M}$ una periodicidad de π y, por simetría, un rango efectivo de $[0, \pi/2]$ para β . El Hamiltoniano de coste, por su parte, admite un rango más amplio de $[0, \pi]$ para γ debido a la mayor variedad de valores propios introducidos por los coeficientes del problema. La fijación de la semilla aleatoria garantiza, una vez más, la reproducibilidad determinista del experimento.

5.6.4. Ejecución del bucle y registro de la trayectoria

La integración de los tres elementos anteriores (**Estimator**, **COBYLA**, e inicialización) configura el bucle híbrido completo. La función **minimize** de **scipy.optimize** orquesta la iteración: en cada paso, **COBYLA** propone un nuevo vector de parámetros, la función **objetivo** ejecuta el circuito QAOA con dichos parámetros y devuelve el valor esperado del Hamiltoniano, y **COBYLA** utiliza ese valor para actualizar su modelo lineal interno y proponer el siguiente candidato. El bucle termina cuando se alcanza el número máximo de iteraciones, cuando el tamaño del simplejo cae por debajo de un umbral de tolerancia, o cuando la mejora marginal entre iteraciones se considera despreciable. La lista **historial** acumula durante toda la ejecución la trayectoria completa de valores esperados evaluados. Esta traza constituye la prueba visual de la convergencia del algoritmo y permitirá, en la Sección 6.4, construir el gráfico de convergencia característico de los algoritmos variacionales, en el que se representa la energía esperada en función del número de iteración junto con el envolvente del mínimo acumulado.

Al término del bucle, el vector **mejores_params** contiene la mejor configuración de parámetros encontrada a lo largo de toda la trayectoria. Esta configuración es la que se utilizará en la siguiente etapa para realizar el muestreo masivo del estado cuántico final y extraer la solución combinatoria al proble-

ma de recuperación de flota, procedimiento que se detalla en la Sección 5.7.

5.7. Extracción y validación de la solución

Al finalizar el bucle híbrido descrito en la Sección 5.6, el vector `mejores_params` contiene la mejor configuración de parámetros variacionales encontrada durante la trayectoria de optimización. Sin embargo, este resultado no constituye todavía una solución al problema de recuperación de flota: lo que se ha optimizado es el valor esperado del Hamiltoniano, una magnitud escalar que resume estadísticamente el estado cuántico, no una configuración concreta de las variables binarias. La extracción de la solución combinatoria requiere una etapa adicional de muestreo del estado cuántico preparado por el circuito con los parámetros óptimos, seguida de la decodificación de las configuraciones obtenidas y de la selección, entre las factibles, de la de menor coste. Esta sección detalla las tres fases de este procedimiento: muestreo mediante la primitiva `Sampler`, decodificación del *bitstring* y selección de la solución factible de menor coste.

5.7.1. Muestreo del estado cuántico óptimo

La extracción de configuraciones combinatorias a partir de un estado cuántico se realiza mediante mediciones repetidas en la base computacional. Cada medición colapsa el estado en una cadena binaria (*bitstring*), y la frecuencia relativa de cada cadena tras un número suficientemente alto de mediciones aproxima la distribución de probabilidad subyacente al estado cuántico. En Qiskit, esta operación se delega a la primitiva `SamplerV2`.

```
1 sampler = Sampler(mode=backend)
2 sampler.options.default_shots = 20000
3 sampler.options.simulator = {"seed_simulator": 42}
4 circuito_medicion = circuito_transpilado.assign_parameters(
    mejores_params)
5 job = sampler.run([circuito_medicion])
6 conteos = job.result()[0].data.meas.get_counts()
```

Listado 5.16: Configuración del muestreador y ejecución del muestreo.

El número de *shots* empleado en esta fase, fijado en 20.000, es sustancialmente

superior al utilizado en la fase de optimización (8.192). Esta diferencia responde a un cambio en el objetivo del muestreo: durante el bucle variacional, la primitiva `Estimator` busca una estimación precisa de un escalar (el valor esperado del Hamiltoniano), para lo cual 8.192 *shots* resultan suficientes; en cambio, en esta fase de extracción de la solución, el objetivo es obtener una muestra amplia de configuraciones de la distribución preparada por el circuito. Un número elevado de mediciones aumenta la probabilidad de que el óptimo, si el estado preparado le asigna una probabilidad apreciable, aparezca al menos una vez en el conteo, que es la única condición necesaria para que pueda ser seleccionado en las fases siguientes.

La asignación de los parámetros óptimos al circuito mediante `assign_parameters` produce un circuito completamente determinista (sin parámetros libres) cuya ejecución repetida bajo el muestreador genera la distribución empírica de configuraciones binarias. El objeto `conteos` resultante es un diccionario que asocia cada *bitstring* observado con el número de veces que ha sido medido.

5.7.2. Decodificación del bitstring

Cada *bitstring* devuelto por el muestreador es una cadena de bits cuya longitud coincide con el número de variables del problema y cuya interpretación requiere dos consideraciones técnicas. La primera es la convención de ordenación: Qiskit emplea por defecto la convención *little-endian*, en la que el bit menos significativo del registro cuántico corresponde al qubit de índice cero y aparece en la posición más a la derecha del *bitstring*. Como en este trabajo las variables binarias se han declarado en el orden definido en la Sección 5.2 (primero las variables de cancelación z_i , después las variables de asignación x_{itk}), la decodificación correcta requiere invertir el orden de los bits antes de mapearlos a los nombres de variable.

```

1 nombres = [v.name for v in mod.variables]
2 for bs, freq in conteos_ordenados[:10]:
3     vec = [int(b) for b in bs[::-1]]
4     # vec[i] es ahora el valor asignado a nombres[i]

```

Listado 5.17: Decodificación del bitstring a vector de asignación.

La segunda consideración afecta a la consistencia entre el *bitstring* obtenido y el modelo original. El operador sobre el que se ha ejecutado QAOA es el Hamiltoniano de Ising derivado de la formulación QUBO, en la que las restricciones del problema están absorbidas en forma de penalizaciones cuadráticas. Esto implica que las configuraciones obtenidas del muestreo no son necesariamente factibles en el problema original: aunque la penalización está calibrada para que las soluciones infactibles tengan una energía elevada y, por tanto, una probabilidad reducida, el muestreo puede incluir alguna de ellas. Por este motivo, la decodificación va seguida de una verificación explícita de factibilidad antes de evaluar el coste.

5.7.3. Selección de la solución factible de menor coste

La validación de cada configuración candidata se realiza mediante el método `is_feasible` del objeto `QuadraticProgram` original (no del QUBO), que evalúa todas las restricciones lineales declaradas en la Sección 5.2 y devuelve un valor booleano. Sobre las configuraciones que resultan factibles se evalúa el coste mediante `objective.evaluate` aplicado al modelo original, lo que garantiza que el valor calculado corresponde a la función objetivo sin penalizaciones, y se conserva la de menor coste.

```
1 conteos_ordenados = sorted(  
2     conteos.items(),  
3     key=lambda kv: kv[1],  
4     reverse=True  
5 )  
6 mejor_sol = None  
7 mejor_coste = float('inf')  
8 for bs, _ in conteos_ordenados[:500]:  
9     vec = [int(b) for b in bs[::-1]]  
10    if mod.is_feasible(vec):  
11        c = mod.objective.evaluate(vec)  
12        if c < mejor_coste:  
13            mejor_coste = c
```

14

```
mejor_sol = bs
```

Listado 5.18: Selección de la mejor solución factible obtenida en el muestreo.

El criterio de selección es, por tanto, el coste objetivo entre las configuraciones factibles, y no la frecuencia con que cada una ha sido muestreada: basta con que el óptimo haya aparecido al menos una vez en el conteo para que sea seleccionado. En la implementación, el recorrido se limita a las 500 configuraciones distintas más frecuentes por simple eficiencia, ya que la cola de la distribución está poblada por configuraciones que aparecen una sola vez y que, en su mayoría, son infactibles; en el caso reducido analizado en este trabajo, esta ventana contiene con holgura el óptimo del problema. Conceptualmente, el procedimiento equivale a recorrer todas las configuraciones distintas obtenidas en los 20.000 disparos y quedarse con la factible de menor coste.

5.7.4. Interpretación de la solución final

Una vez identificado el *bitstring* de menor coste, su interpretación dentro del escenario reducido es directa: cada variable con valor 1 representa una decisión concreta de la solución del modelo simplificado de recuperación. Las variables x_{itk} activas indican qué vuelos se operan, en qué banda temporal y con qué aeronave; las variables z_i activas indican qué vuelos se cancelan. El coste asociado a la solución se descompone, por tanto, en la suma de los costes de retraso de los vuelos operados con desplazamiento temporal y los costes de cancelación de los vuelos no recuperados.

```
1 vec = [int(b) for b in mejor_sol[::-1]]
2 for nombre, val in zip(nombres, vec):
3     if val == 1:
4         print(f" {nombre} = 1")
```

Listado 5.19: Interpretación de la solución óptima.

Este vector de decisiones constituye el resultado final del flujo QAOA. No obstante, su mera obtención no es suficiente para validar la corrección del

enfoque cuántico: es necesario verificar que la solución encontrada por el algoritmo coincide con el óptimo global del problema, descartando la posibilidad de que el algoritmo haya convergido a una solución subóptima dentro del espacio factible. Esta validación se aborda mediante la implementación de un *solver* clásico de referencia que resuelve el problema por enumeración exhaustiva del espacio de configuraciones, procedimiento que se detalla en la Sección 5.8. El análisis comparativo de los resultados de ambos enfoques y la discusión sobre la calidad de la convergencia se presentan en el Capítulo 6.

5.8. Validación clásica del modelo

El procedimiento descrito en la Sección 5.7 concluye con la identificación, por parte del algoritmo QAOA, de una configuración factible de coste mínimo dentro del conjunto de estados muestreados. Sin embargo, este resultado por sí solo no constituye una prueba de la corrección del enfoque cuántico: el algoritmo podría haber convergido a un mínimo local del paisaje energético, a una solución sub-óptima dentro del espacio factible, o a una configuración que minimice la función objetivo penalizada sin coincidir con el óptimo del problema original. Para descartar estas posibilidades y validar empíricamente que QAOA recupera la solución correcta del caso reducido, se ha desarrollado un solver clásico de referencia que resuelve el mismo problema por enumeración exhaustiva del espacio de configuraciones. Esta sección detalla el diseño y la justificación de dicho algoritmo, así como su rol como instrumento de validación de la prueba de concepto.

5.8.1. Justificación del enfoque de fuerza bruta

La elección de un solver basado en enumeración exhaustiva, en lugar de un optimizador clásico más sofisticado como un algoritmo de ramificación y poda o un solver de programación entera mixta, responde a tres consideraciones metodológicas. En primer lugar, la enumeración exhaustiva proporciona la garantía teórica más fuerte posible sobre la solución obtenida: al evaluar todas las configuraciones del espacio binario, el óptimo identificado es necesariamente el óptimo global del problema, no una aproximación. Esta propiedad es indispensable cuando el objetivo es validar la convergencia de un algoritmo aproximado como QAOA, ya que cualquier solver heurístico introduciría una segunda fuente de incertidumbre en la comparación.

En segundo lugar, el tamaño del problema reducido lo hace computacionalmente viable. El espacio de configuraciones contiene $2^{15} = 32,768$ combinaciones, una magnitud que un equipo de cómputo convencional puede procesar en segundos. Esta característica del escenario experimental, que en otros con-

textos sería una limitación, se convierte aquí en una ventaja metodológica: permite establecer una referencia absoluta del óptimo sin recurrir a aproximaciones.

En tercer lugar, la naturaleza de la validación que se persigue es binaria. No interesa medir la *calidad relativa* de la solución de QAOA frente a un benchmark sub-óptimo, sino determinar si QAOA recupera o no la solución globalmente óptima. Para esta pregunta cuya respuesta es afirmativo o negativo, un solver exacto es la herramienta apropiada; un solver heurístico podría inducir conclusiones engañosas si su propia aproximación fuera peor que la del algoritmo cuántico.

5.8.2. Diseño del solver y pre-construcción de estructuras

Un enfoque ingenuo para implementar la enumeración consistiría en iterar sobre todas las configuraciones binarias y, para cada una, invocar los métodos del objeto `QuadraticProgram` (`is_feasible` y `objective.evaluate`) para verificar factibilidad y calcular coste. Esta aproximación, sin embargo, es demasiado lenta debido al sobrecoste de las llamadas a los métodos genéricos de la librería, que internamente realizan validaciones, conversiones de tipo y operaciones de búsqueda en estructuras de datos no optimizadas para evaluación repetida masiva.

Para evitar este sobrecoste, el solver implementado realiza una fase previa de pre-construcción en la que las restricciones del modelo se traducen a estructuras de datos nativas de Python optimizadas para evaluación rápida. Esta fase se ejecuta una sola vez, antes del bucle de enumeración, y produce tres estructuras principales: una lista de listas de índices que representa las restricciones de unicidad agrupadas por vuelo, una lista de pares de índices que representa las restricciones de continuidad como pares de variables mutuamente incompatibles, y un vector de coeficientes que asocia a cada variable su coste lineal.

```

1 unicidad = []
2 for i in vuelos_lista:
3     lst = [idx[f"z_{i}"]]
4     for t in range(num_bandas):
5         for k in aeronaves:
6             lst.append(idx[f"x_{i}_t{t}_{k}"])
7     unicidad.append(lst)
8 continuidad = []
9 for k in aeronaves:
10    for i1, i in enumerate(vuelos_lista):
11        for i2, j in enumerate(vuelos_lista):
12            if i1 >= i2:
13                continue
14            for t_i in range(num_bandas):
15                for t_j in range(num_bandas):
16                    # [... cálculo de intervalos de ocupación ...]
17                    if (ini_i < fin_j) and (ini_j < fin_i):
18                        continuidad.append(
19                            (idx[f"x_{i}_t{t_i}_{k}"],
20                             idx[f"x_{j}_t{t_j}_{k}"])
21                        )

```

Listado 5.20: Pre-construcción de estructuras de evaluación rápida.

La lógica geométrica que genera estas estructuras es idéntica a la utilizada en la Sección 5.2 para declarar las restricciones del `QuadraticProgram`, lo que garantiza que el solver clásico y el modelo cuántico resuelven exactamente el mismo problema. Esta equivalencia es esencial: si los dos enfoques operaran sobre formulaciones ligeramente distintas, cualquier discrepancia en sus soluciones sería interpretable como un error de modelado y no como una característica del algoritmo.

5.8.3. Enumeración y selección del óptimo

Con las estructuras pre-construidas, el bucle de enumeración recorre el producto cartesiano $\{0, 1\}^n$ y, para cada configuración binaria, verifica las restricciones mediante operaciones aritméticas elementales sobre las estructuras pre-construidas. Una configuración se descarta tan pronto como se detecta el incumplimiento de cualquier restricción, lo que permite saltar al siguiente

te candidato sin completar la evaluación. Esta estrategia de evaluación con cortocircuito reduce sustancialmente el tiempo medio por configuración.

```

1 for bits in itertools.product([0, 1], repeat=n_vars):
2     ok = True
3     for lst in unicidad:
4         if sum(bits[i] for i in lst) != 1:
5             ok = False
6             break
7     if not ok:
8         continue
9     for a, b in continuidad:
10        if bits[a] + bits[b] > 1:
11            ok = False
12            break
13    if not ok:
14        continue
15    factibles += 1
16    c = sum(bits[i] * coste_var[i] for i in range(n_vars))
17    if c < mejor_coste:
18        mejor_coste = c
19        mejor_bits = bits

```

Listado 5.21: Bucle principal de enumeración y selección del óptimo.

El contador `factibles` acumula el número total de configuraciones que satisfacen todas las restricciones del modelo, una métrica relevante en sí misma: caracteriza la densidad del espacio factible dentro del espacio total de configuraciones y proporciona el contexto estructural necesario para interpretar tanto el comportamiento de QAOA como la calibración empírica del coeficiente de penalización discutida en la Sección 5.4. El registro `mejor_bits` mantiene la configuración factible de menor coste encontrada hasta el momento, que al término del bucle constituye el óptimo global del problema.

5.8.4. Rol del solver en la validación de la prueba de concepto

El solver clásico cumple, en el contexto de este trabajo, una función estrictamente metodológica: no se propone como alternativa operativa al algoritmo

cuántico, sino como instrumento de validación. La comparación entre la solución devuelta por QAOA y el óptimo identificado por el solver permite responder a la pregunta central de la prueba de concepto: ¿es capaz QAOA, con la configuración de hiperparámetros y la calibración de penalizaciones descritas en este capítulo, de recuperar el óptimo global del escenario reducido de recuperación de flota? La respuesta a esta pregunta, así como el análisis cuantitativo de la comparación entre ambos enfoques, se aborda en el Capítulo 6. Conviene señalar que la viabilidad de este esquema de validación está intrínsecamente ligada al tamaño reducido del escenario experimental. Para problemas de la escala del caso de uso teórico de 444 variables, la enumeración exhaustiva sería computacionalmente intratable (2^{444} configuraciones), lo que invalidaría este esquema de comparación directa. En tales situaciones, la validación del algoritmo cuántico requeriría enfoques alternativos basados en solvers heurísticos de alta calidad o en cotas teóricas derivadas de relajaciones del problema. La adopción del enfoque de fuerza bruta es, por tanto, una elección coherente con la estrategia de reducción experimental descrita en el Capítulo 6: el escenario se ha dimensionado precisamente para que admita una validación clásica rigurosa, condición necesaria para que la prueba de concepto sea concluyente. Con la implementación de este solver de referencia se completa el desarrollo tecnológico del trabajo. El Capítulo 6 aborda a continuación la ejecución experimental sobre el escenario reducido, la presentación de los resultados obtenidos por ambos enfoques y la discusión sobre la calidad de la convergencia, el comportamiento del circuito en el simulador y la proyección sobre la viabilidad del enfoque en hardware real.

Capítulo 6. Análisis de resultados, validación y discusión

6.1. Estrategia experimental y dimensionado del caso de uso

La validación experimental de un algoritmo cuántico variacional como QAOA requiere una decisión metodológica previa que condiciona críticamente la interpretación de los resultados: la elección del tamaño del problema sobre el que se ejecuta el algoritmo. Esta elección no es trivial, ya que existe un compromiso fundamental entre dos exigencias contrapuestas. Por un lado, el escenario debe conservar rasgos estructurales suficientes para que las conclusiones obtenidas sean relevantes para interpretar el problema operativo de fondo; por otro, debe ser suficientemente reducido para que su ejecución sea computacionalmente viable en los recursos disponibles y para que la validación contra un *benchmark* exacto sea técnicamente posible.

Este compromiso adquiere especial relevancia en el contexto de la computación cuántica en simuladores clásicos, donde el coste de simulación crece exponencialmente con el número de qubits y la profundidad del circuito. La estrategia adoptada en este trabajo ha consistido en explorar tres niveles sucesivos de dimensionado, cada uno con su propia justificación y resultado, hasta identificar el punto en el que el modelo es a la vez ejecutable, validable y representativo. Esta sección documenta dicha exploración y explica las razones que conducen al escenario reducido sobre el que se realiza la validación empírica.

6.1.1. Inviabilidad de la simulación a escala objetivo e intermedia

El escenario operativo de referencia, formulado en el Capítulo 4, da lugar a un modelo de 444 variables binarias y un espacio de configuraciones de 2^{444} estados. Esta magnitud es inabordable mediante simulación clásica de vector de estado: la representación del estado cuántico de 444 qubits exigiría un vector complejo de 2^{444} componentes, físicamente irrealizable con cualquier capacidad de memoria existente. No se trata de una limitación específica de este trabajo, sino del estado actual de la simulación cuántica clásica; Kaushik et al. (2025) sitúan típicamente por debajo de los 30 qubits los escenarios validables por esta vía, recurriendo para tamaños mayores a *hardware* cuántico real.

Esta barrera se manifiesta de forma abrupta incluso a escalas muy inferiores a la objetivo. La evaluación de la función objetivo mediante la primitiva **Estimator** sobre un escenario intermedio de aproximadamente 50 variables se interrumpe con un error de asignación de memoria: el simulador requiere una reserva de 68 719 476 736 megabytes de RAM, equivalentes a unos 65 500 terabytes, frente a los 16,1 gigabytes disponibles en la estación de trabajo utilizada. Este dato cuantifica el carácter exponencial del coste de simulación: el salto desde un régimen viable hasta uno computacionalmente intratable se produce en una ventana muy estrecha, en torno a los 40-50 qubits.

En consecuencia, el escenario de 444 variables se mantiene como caso de uso teórico y referencia operativa, pero la validación empírica del modelo exige reducir el problema a una escala simulable en los recursos disponibles, manteniendo los elementos estructurales necesarios para la prueba de concepto.

6.1.2. Escenario reducido: 15 variables binarias

El escenario finalmente adoptado para la validación experimental corresponde a una versión sustancialmente reducida del caso operativo, manteniendo no

obstante los elementos estructurales necesarios para la prueba de concepto: pluralidad de vuelos, pluralidad de aeronaves, bandas temporales de retraso, restricciones de unicidad, restricciones de continuidad por solapamiento de intervalos de ocupación, y posibilidad de cancelación. La reducción afecta principalmente al tamaño de los conjuntos involucrados, manteniendo la lógica básica de la formulación matemática empleada en el trabajo.

Tal como se explica en la sección Sección 4.5, el escenario reducido contempla tres vuelos consecutivos en una ruta MAD-BCN-CDG-AMS, atendidos por dos aeronaves operativas (AC2 y AC3, asumiendo que la AC1 queda inhabilitada), con dos bandas temporales por vuelo (operación puntual o retraso de quince minutos) y la opción de cancelación. La combinación de estos parámetros genera tres variables de cancelación más doce variables de asignación (tres vuelos por dos bandas por dos aeronaves), totalizando quince variables binarias y un espacio de configuraciones de $2^{15} = 32\,768$ estados.

Esta magnitud del problema satisface simultáneamente las tres condiciones que motivan la estrategia experimental:

- Es simulable en *hardware* clásico convencional sin requerimientos de memoria excepcionales, permitiendo la ejecución completa del bucle híbrido QAOA en tiempos del orden de decenas de segundos.
- Es enumerable de forma exhaustiva mediante el *solver* clásico descrito en la Sección 5.8, lo que habilita la comparación directa entre el resultado de QAOA y el óptimo global garantizado del problema.
- Preserva la complejidad combinatoria esencial de la formulación simplificada: el espacio de soluciones factibles es esparcido (como se cuantificará en la Sección 6.3), existen conflictos temporales efectivos entre vuelos asignados a una misma aeronave, y el algoritmo debe decidir asignaciones no triviales entre operación, retraso y cancelación.

Conviene subrayar que la reducción al escenario de 15 variables responde a una elección metodológica que prima el rigor de la validación frente a la escala, condicionada por las capacidades de simulación clásica de circuitos cuánticos. Como señala Sipos et al. (2025) en su análisis sobre *benchmarks*

variacionales para optimización aeronáutica, los estudios de prueba de concepto sobre QAOA aplicado a problemas industriales se sitúan habitualmente en el rango de 10 a 30 variables. Esta acotación no se debe a una limitación del *solver* clásico de referencia (que, por enumeración exhaustiva, resuelve sin dificultad problemas de este tamaño e incluso mayores), sino al coste de simular el propio algoritmo cuántico: como se ha cuantificado en la subsección anterior, la simulación de vector de estado se vuelve intratable en una ventana estrecha en torno a los 40-50 qubits. Trabajar por debajo de ese umbral es lo que permite, simultáneamente, ejecutar QAOA en el simulador y disponer de una referencia clásica exacta contra la que contrastar su resultado. La validación rigurosa de la convergencia de QAOA exige, por tanto, operar en este régimen reducido como paso previo y necesario a cualquier escalado posterior sobre *hardware* real.

Sobre este escenario reducido se ejecuta el algoritmo QAOA con la configuración detallada en el Capítulo 5, cuyos parámetros experimentales concretos y resultados se presentan en las secciones siguientes. La Sección 6.2 consolida la configuración utilizada en una tabla resumen y enmarca el contexto experimental antes de pasar al análisis de los resultados.

6.2. Configuración experimental del caso de 15 variables

Una vez justificada en la Sección 6.1 la elección del escenario reducido como caso de validación, esta sección consolida la configuración experimental completa sobre la que se ejecutan tanto el algoritmo QAOA como el *solver* clásico de referencia. El objetivo es proporcionar al lector una visión integrada y reproducible de todos los parámetros relevantes del experimento, agrupados en bloques temáticos: el escenario operativo, la formulación QUBO, la configuración del circuito cuántico, los parámetros del bucle de optimización híbrida, el muestreo final y las condiciones de reproducibilidad.

6.2.1. Escenario operativo

El escenario contempla una ventana de recuperación de tres horas, comprendida entre las 14:00 y las 17:00, durante la cual deben operarse tres vuelos consecutivos en una red europea. La avería de la aeronave AC1 a las 14:00 deja al sistema con dos aeronaves operativas (AC2 y AC3), que deben absorber los tres vuelos programados o cancelar aquellos que generen conflictos de continuidad irresolubles. Cada vuelo dispone de dos bandas temporales factibles: su hora de salida original (banda $t = 0$) y un retraso de quince minutos (banda $t = 1$). Los datos completos del programa de vuelos se recogen en el Tabla 6.1.

Cuadro 6.1: Programa de vuelos del escenario reducido

Vuelo	Origen	Destino	Salida	Duración	Llegada
101	MAD	BCN	14:00	1h 15min	15:15
201	BCN	CDG	14:30	1h 45min	16:15
301	CDG	AMS	15:00	1h 30min	16:30

El tiempo de rotación entre vuelos consecutivos operados por la misma ae-

ronave se fija en treinta minutos, valor representativo para escalas cortas en aeropuertos europeos. Este parámetro condiciona la generación automática de las restricciones de continuidad, ya que define el intervalo de ocupación efectiva de una aeronave $[t_{\text{salida}}, t_{\text{salida}} + d + r]$ que no puede solaparse con la operación de otro vuelo.

6.2.2. Estructura del problema QUBO

La aplicación de la formulación matemática descrita en el Capítulo 4 al escenario anterior genera un modelo con quince variables binarias: tres variables de cancelación z_i (una por vuelo) y doce variables de asignación x_{itk} (tres vuelos por dos bandas por dos aeronaves). El espacio de configuraciones contiene $2^{15} = 32\,768$ estados, sobre el que el modelo impone tres restricciones de unicidad (una por vuelo) y un conjunto de restricciones de continuidad cuyo número exacto depende de los solapamientos detectados automáticamente por la lógica geométrica del modelo.

La conversión a formulación QUBO se realiza con el coeficiente de penalización empíricamente calibrado en la Sección 5.4. Los valores económicos del caso reducido deben interpretarse como una escala normalizada de validación, no como una estimación monetaria absoluta del coste real de una disrupción. Su función es preservar la jerarquía del problema, $C_{\text{CANCEL}} \gg C_{\text{DELAY}}$: retrasar un vuelo debe resultar menos costoso que cancelarlo, y violar una restricción debe resultar energéticamente menos atractivo que cualquier solución factible razonable.

6.2.3. Parámetros completos del experimento

La configuración experimental completa, incluyendo tanto los parámetros del modelo como los del algoritmo QAOA y del bucle de optimización, se sintetiza en el Tabla 6.2. Esta tabla constituye la referencia única para la reproducibilidad del experimento; los valores económicos que recoge corresponden a la parametrización normalizada del caso reducido.

Cuadro 6.2: Configuración experimental completa del caso reducido de 15 variables

Bloque	Parámetro	Valor
Escenario operativo	Número de vuelos	3
	Aeronaves operativas	2 (AC2, AC3)
	Bandas temporales por vuelo	2
	Intervalo entre bandas	15 min
	Tiempo de rotación	30 min
Modelo QUBO	Variables binarias	15
	Coste por banda de retraso (C_{DELAY})	50 €
	Coste de cancelación (C_{CANCEL})	500 €
	Coeficiente de penalización (P)	3 000 €
Circuito QAOA	Número de repeticiones (P_{reps})	5
	<i>Backend</i> de simulación	AerSimulator
	Nivel de optimización del transpilador	3
Bucle híbrido	Optimizador clásico	COBYLA
	Máximo de iteraciones	800
	Tamaño inicial del simplejo (rhobeg)	0,3
	<i>Shots</i> por evaluación (<i>Estimator</i>)	8 192
Muestreo final	<i>Shots</i> de muestreo (<i>Sampler</i>)	20 000
	Ventana de búsqueda de factibles	<i>Top</i> 500
Reproducibilidad	Semilla NumPy	42
	Semilla simulador	42

La fijación explícita de las semillas aleatorias tanto en la inicialización de los parámetros variacionales como en el simulador cuántico garantiza la reproducibilidad determinista del experimento: cualquier ejecución del código fuente con esta configuración debe producir los mismos resultados numéricos hasta el último dígito significativo.

Con esta configuración consolidada, las siguientes secciones presentan los resultados experimentales propiamente dichos. La Sección 6.3 reporta los resultados del *solver* clásico, que establecen la referencia absoluta del problema. La Sección 6.4 documenta el comportamiento del algoritmo QAOA. Final-

mente, la Sección 6.5 contrasta ambos enfoques y discute las implicaciones de la comparación.

6.3. Resultados del *solver* clásico de referencia

La ejecución del *solver* clásico descrito en la Sección 5.8 sobre el escenario reducido configurado en la Sección 6.2 produce el conjunto completo de soluciones factibles del problema y permite identificar, mediante enumeración exhaustiva, el óptimo global del modelo. Esta sección presenta los resultados obtenidos por el *solver* y los analiza desde tres perspectivas complementarias: la caracterización del espacio de soluciones, la identificación del óptimo y su interpretación operativa, y el papel de esta solución como referencia absoluta para la posterior validación del algoritmo QAOA.

6.3.1. Caracterización del espacio de soluciones factibles

La enumeración exhaustiva del espacio de configuraciones del modelo, compuesto por $2^{15} = 32\,768$ estados binarios, identifica un total de **37 soluciones factibles**, es decir, configuraciones que satisfacen simultáneamente las tres restricciones de unicidad y todas las restricciones de continuidad generadas automáticamente a partir de los solapamientos temporales del programa de vuelos. Esta cifra implica una **densidad del espacio factible del 0,11 por ciento**, una proporción notablemente reducida que tiene implicaciones directas sobre el comportamiento esperado del algoritmo cuántico.

Esta densidad extremadamente baja es la propiedad estructural que justifica retrospectivamente la calibración empírica del coeficiente de penalización discutida en la Sección 5.4. Cuando el espacio factible representa menos del 0,2 por ciento del espacio total, la mayor parte del paisaje energético explorado por QAOA corresponde a configuraciones que violan al menos una restricción, lo que exige coeficientes de penalización suficientemente grandes para garantizar que el mínimo energético del Hamiltoniano de coste coincide con el óptimo factible del problema original. El valor $p = 3\,000$ adoptado en este trabajo, sustancialmente superior al rango teórico derivado de la cota

de factibilidad estricta, es coherente con esta característica del espacio de búsqueda y proporciona evidencia empírica adicional a favor del enfoque de calibración adaptativa descrito en la Sección 5.4.

El coste computacional de la enumeración exhaustiva sobre los 32 768 estados resulta inferior a un segundo en la estación de trabajo utilizada, lo que confirma la idoneidad del enfoque de fuerza bruta como mecanismo de validación para esta escala del problema, según se anticipó en la Sección 5.8.

6.3.2. Óptimo global del problema

Entre las 37 configuraciones factibles, el *solver* identifica una única solución de coste mínimo, con un **coste objetivo de 500 €**. La asignación de variables correspondiente a este óptimo se presenta en el Tabla 6.3.

Cuadro 6.3: Asignación óptima identificada por el *solver* clásico

Variable activa	Significado operativo	Coste asociado
$x_{101,t=0,AC3} = 1$	Vuelo 101 operado puntualmente por AC3	0 €
$z_{201} = 1$	Vuelo 201 cancelado	500 €
$x_{301,t=0,AC2} = 1$	Vuelo 301 operado puntualmente por AC2	0 €
Coste total		500 €

6.3.3. Interpretación operativa del óptimo

La estructura de la solución óptima admite una interpretación directa dentro del modelo temporal simplificado y confirma la coherencia interna de las restricciones implementadas. Tras la avería de AC1 a las 14:00, el sistema queda con dos aeronaves disponibles (AC2 y AC3) que deben absorber los tres vuelos del programa. La distribución óptima asigna los dos vuelos compatibles temporalmente a cada una de las aeronaves operativas: el vuelo 101 (MAD-BCN, 14:00-15:15) se asigna a AC3, y el vuelo 301 (CDG-AMS, 15:00-16:30) se asigna a AC2. Ambos vuelos se operan puntualmente, sin incurrir en coste alguno de retraso.

El vuelo 201 (BCN-CDG, 14:30-16:15), por el contrario, queda cancelado. La razón estructural de esta decisión se identifica a partir de los intervalos temporales considerados por el modelo. El vuelo 201 presenta solapamientos de ocupación con las alternativas que permitirían operar simultáneamente los otros vuelos dentro de la ventana reducida. Bajo las restricciones de continuidad temporal implementadas, ninguna asignación de AC2 o AC3 al vuelo 201 permite conservar una solución factible de menor coste. La cancelación de este vuelo es, por tanto, la alternativa óptima dentro del modelo simplificado.

El coste total de 500 € corresponde exclusivamente a la penalización por cancelación del vuelo 201, sin contribución alguna de coste por retraso. Esta solución representa el equilibrio óptimo del modelo simplificado: dado que las alternativas de operación del vuelo 201 generan conflictos temporales con el resto de decisiones factibles, la mejor solución dentro de la formulación considerada consiste en aceptar la cancelación de un único vuelo y preservar la operación puntual de los dos restantes.

6.4. Resultados del algoritmo QAOA

Esta sección documenta el comportamiento del algoritmo QAOA durante su ejecución sobre el escenario reducido, atendiendo a las dos fases del procedimiento descrito en el Capítulo 5: la convergencia del bucle híbrido de optimización de parámetros (Sección 5.6) y la extracción de la solución mediante el muestreo del estado cuántico óptimo (Sección 5.7). El análisis comparativo con la referencia clásica establecida en la Sección 6.3 se reserva para la Sección 6.5.

6.4.1. Convergencia del bucle híbrido

El bucle de optimización híbrida, ejecutado con la configuración detallada en la Sección 6.2, alcanzó la convergencia tras 79 evaluaciones de la función objetivo, con un tiempo total de ejecución de 27,32 segundos en el simulador local. El valor mínimo del valor esperado del Hamiltoniano registrado a lo largo de la trayectoria fue de $-26\,183,73$ unidades de energía, alcanzado en la iteración 72. La Figura 6.1 representa la evolución de este valor esperado en función del número de iteración del optimizador COBYLA.

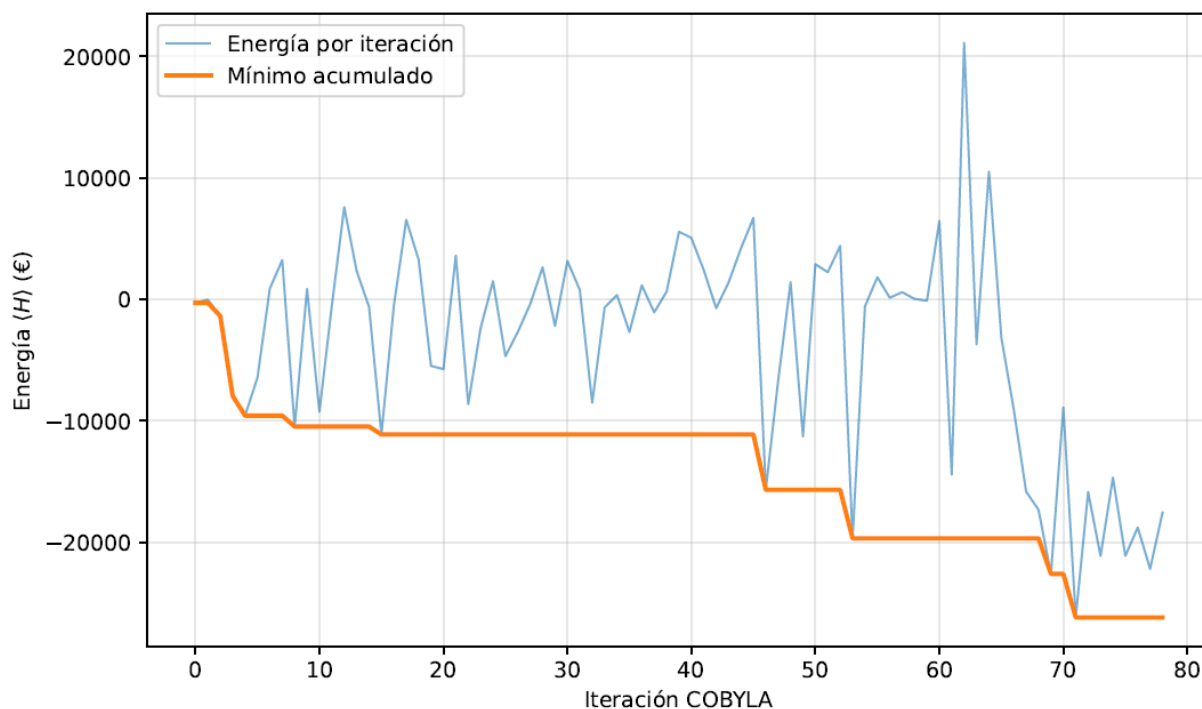


Figura 6.1: Convergencia del bucle híbrido QAOA. La línea de trazo fino representa el valor esperado del Hamiltoniano evaluado en cada iteración del optimizador COBYLA; la línea de trazo grueso representa el mínimo acumulado a lo largo de la trayectoria.

La gráfica ilustra dos comportamientos característicos de los algoritmos variacionales cuánticos. En primer lugar, la traza de la energía evaluada en cada iteración (línea fina) presenta oscilaciones pronunciadas, alternando valores positivos y negativos a lo largo de toda la ejecución. Este patrón errático no es indicativo de un mal funcionamiento, sino que refleja la naturaleza exploratoria del optimizador COBYLA, que evalúa puntos del espacio de parámetros distribuidos alrededor de su simplejo de trabajo, incluyendo configuraciones de alta energía necesarias para caracterizar localmente la superficie de coste. En segundo lugar, el envolvente del mínimo acumulado (línea gruesa) muestra un descenso escalonado y monótono: una vez que el optimizador identifica una región de menor energía, esta queda registrada como mejor solución y no se pierde, aun cuando las iteraciones posteriores exploren regiones peores.

El perfil de descenso revela una estructura por etapas. Tras una caída inicial rápida en las primeras iteraciones, que lleva el mínimo desde valores próximos a cero hasta el entorno de $-11\,000$ en torno a la iteración 16, el algoritmo

atraviesa una fase de meseta prolongada en la que no logra mejorar la mejor solución encontrada durante casi treinta iteraciones. A partir de la iteración 47, el optimizador encadena una serie de saltos sucesivos de mejora ($-15\,675$ en la iteración 47, $-19\,669$ en la 54, $-22\,593$ en la 70) que culminan en el valor final de $-26\,183,73$ en la iteración 72. Este patrón de mesetas seguidas de mejoras abruptas es coherente con el comportamiento reportado por Sipos et al. (2025) para QAOA aplicado a problemas combinatorios de aviación, en los que la superficie de coste presenta regiones planas separadas por barreras que el optimizador debe sortear.

6.4.2. Interpretación de la escala energética

Conviene precisar la interpretación del valor mínimo de energía alcanzado, $-26\,183,73$, para evitar una confusión frecuente. Este valor no representa el coste operativo de la solución, sino el valor esperado del Hamiltoniano de Ising, una magnitud que incorpora dos contribuciones ajenas al coste objetivo del problema original. Por un lado, el cambio de variable $x_i = (1 - z_i)/2$ aplicado en el mapeo a Ising (Sección 5.4) introduce un término constante de *offset*, que en este caso asciende a 50 400 unidades. Por otro, las penalizaciones cuadráticas absorbidas en la función objetivo durante la conversión a QUBO contribuyen también al valor del Hamiltoniano para las distintas configuraciones.

Por este motivo, la energía del Hamiltoniano y el coste operativo de la solución son magnitudes distintas que no deben compararse directamente. El coste operativo real de la solución se recupera únicamente tras decodificar el *bitstring*, verificar su factibilidad y evaluar la función objetivo original sin penalizaciones, según el procedimiento descrito en la Sección 5.7. El valor de energía cumple, en el contexto del bucle de optimización, una función estrictamente instrumental: es la cantidad que el optimizador minimiza para guiar la búsqueda de los parámetros variacionales óptimos, no una métrica directa de la calidad operativa de la solución.

6.4.3. Extracción de la solución mediante muestreo

Una vez fijados los parámetros variacionales óptimos, se ejecutó el muestreo del estado cuántico resultante mediante la primitiva **Sampler** con 20 000 disparos, según el procedimiento de la Sección 5.7. El muestreo produce una distribución de probabilidad sobre los estados de la base computacional, de la cual se extraen las configuraciones candidatas ordenadas por frecuencia de aparición. El Tabla 6.4 recoge los diez estados más probables de la distribución, indicando para cada uno su frecuencia de aparición y su condición de factibilidad junto con el coste operativo en los casos factibles.

Cuadro 6.4: Diez estados más probables de la distribución muestreada por QAOA

Pos.	<i>Bitstring</i>	Frecuencia	Estado
1	000000000000111	296	Factible (1 500 €)
2	110000000000011	144	Infactible
3	000011000000101	131	Infactible
4	000000001100110	124	Infactible
5	001100000000111	87	Infactible
6	000000000000110	86	Infactible
7	000000000000011	85	Infactible
8	000000000000010	79	Infactible
9	000000110000111	79	Infactible
10	000000000000000	78	Infactible

El análisis de la distribución muestreada revela un resultado de relevancia metodológica: el óptimo global del problema, con coste operativo de 500 €, no figura entre los estados más probables del muestreo. El estado más frecuente (frecuencia 296) corresponde a una configuración factible pero subóptima, que cancela los tres vuelos con un coste de 1 500 €; los nueve estados siguientes en orden de probabilidad son todos infactibles, ya que violan alguna de las restricciones de unicidad o continuidad del modelo.

Este hallazgo confirma la necesidad del procedimiento de post-selección descrito en la Sección 5.7. Una estrategia que se limitase a seleccionar el *bitstring* de mayor probabilidad habría devuelto la solución subóptima de 1 500 €. El procedimiento implementado, en cambio, recorre los 500 estados más frecuentes de la distribución, descarta los infactibles mediante el método `is_feasible` y selecciona, entre los factibles, el de menor coste operativo. El óptimo de 500 € aparece en el muestreo con baja frecuencia, fuera de los diez estados más probables, pero su sola presencia en la distribución basta para que el filtrado lo identifique y lo seleccione como solución final. La estructura de variables activas de esta solución y su equivalencia con la referencia clásica se analizan en detalle en la Sección 6.5.

6.5. Comparación cuántica frente a clásica y validación de la prueba de concepto

Las dos secciones anteriores han presentado, de forma independiente, los resultados del *solver* clásico de referencia (Sección 6.3) y los del algoritmo QAOA (Sección 6.4). Esta sección confronta ambos resultados para verificar la validez experimental de este trabajo: ¿es capaz el algoritmo QAOA de recuperar el óptimo global del escenario reducido de recuperación de flota? El análisis de esta equivalencia, sus implicaciones sobre la corrección del modelo y los límites de la conclusión que de ella puede extraerse constituyen el núcleo de esta sección.

6.5.1. Equivalencia de las soluciones

El *solver* clásico, mediante enumeración exhaustiva del espacio de 2^{15} configuraciones, identificó un óptimo global con un coste operativo de 500 €, correspondiente a la cancelación del vuelo 201 y la operación puntual de los vuelos 101 y 301 por las aeronaves AC3 y AC2 respectivamente (Tabla 6.3). El algoritmo QAOA, tras el muestreo del estado cuántico óptimo y el procedimiento de post-selección, recuperó una solución factible con idéntico coste operativo de 500 €. El Tabla 6.5 confronta ambas soluciones variable a variable.

Cuadro 6.5: Comparación de la solución óptima clásica y la solución recuperada por QAOA

Variable	<i>Solver</i> clásico	QAOA
$x_{101,t=0,AC3}$	1	1
z_{201}	1	1
$x_{301,t=0,AC2}$	1	1
Resto de variables	0	0
Coste total	500 €	500 €

6.5.2. Implicaciones de la validación

La recuperación del óptimo global del caso reducido por parte del flujo QAOA, muestreo y posprocesado clásico aporta evidencia sobre tres aspectos críticos del desarrollo, cada uno correspondiente a una etapa distinta del flujo descrito en el Capítulo 5.

En primer lugar, valida la **corrección de la formulación QUBO** para el caso reducido analizado. Si la traducción del modelo con restricciones a la formulación cuadrática sin restricciones hubiera introducido algún error (un signo equivocado, un término de penalización mal construido o una restricción omitida), la correspondencia entre el Hamiltoniano resultante y la formulación original del caso reducido se habría visto alterada, dificultando la recuperación del óptimo observado. La coincidencia exacta confirma que el mapeo desde el `QuadraticProgram` hasta el Hamiltoniano de Ising preserva fielmente la estructura del problema.

En segundo lugar, valida la **calibración del coeficiente de penalización**. Como se discutió en la Sección 5.4, la elección de $P = 3\,000$ se realizó empíricamente, por encima del rango teórico, ante la baja densidad del espacio factible. La recuperación del óptimo factible demuestra que este valor es efectivamente suficiente para garantizar que el mínimo energético del Hamiltoniano corresponde a una solución que respeta todas las restricciones: si la penalización hubiera sido insuficiente, el algoritmo habría favorecido configuraciones de menor coste aparente pero infactibles, y el óptimo factible no habría dominado el espectro.

En tercer lugar, valida que la **profundidad del *ansatz*** seleccionada, $P_{\text{reps}} = 5$, proporciona expresividad suficiente para que el circuito QAOA represente un estado cuántico cuya distribución de probabilidad contiene el óptimo global con probabilidad suficiente. Aunque, como se evidenció en la Sección 6.4, dicho óptimo no es el estado más probable del muestreo, su presencia en la distribución es condición necesaria para que el post-procesado pueda rescatarlo. Una profundidad insuficiente habría hecho necesario un número muy elevado de *shots* para obtenerla.

6.5.3. Alcance y límites de la conclusión

La equivalencia demostrada constituye una validación rigurosa de la *corrección* del enfoque QAOA aplicado al problema de recuperación de flota: el algoritmo, con la formulación y la configuración desarrolladas en este trabajo, recupera el óptimo global del problema en el escenario reducido. Es importante, no obstante, delimitar con precisión el alcance de esta conclusión para evitar sobreinterpretaciones.

En primer lugar, la validación se ha realizado sobre un escenario de 15 variables, dimensionado precisamente para poder realizar la simulación del *hardware* cuántico. La coincidencia con el óptimo clásico demuestra que el algoritmo funciona correctamente a esta escala, pero no constituye, por sí sola, evidencia de que QAOA mantenga esta capacidad en escenarios de mayor tamaño, donde la profundidad del circuito, el ruido del *hardware* y la complejidad del paisaje energético introducen dificultades adicionales.

En segundo lugar, y de forma aún más relevante, este resultado no constituye una demostración de ventaja cuántica. En el escenario reducido, el *solver* clásico de fuerza bruta resuelve el problema en un tiempo inferior al de QAOA, lo que es esperable: para problemas de esta escala, la computación clásica es trivialmente superior. El valor del resultado no reside, por tanto, en una comparación de eficiencia, sino en la demostración de *viabilidad y corrección* del enfoque cuántico como prueba de concepto. La hipótesis de una eventual ventaja cuántica solo sería contrastable en el régimen de escala en el que la simulación clásica deja de ser viable (en torno a las 50 variables, según se documentó en la Sección 6.1), régimen que requiere *hardware* cuántico real y queda fuera del alcance experimental de este trabajo.

En síntesis, la prueba de concepto queda validada en sus propios términos: la cadena completa de modelado, conversión, ejecución cuántica y post-procesado descrita en el Capítulo 5 produce el resultado correcto sobre el escenario reducido. Esta validación, sin embargo, se ha obtenido en un entorno de simulación ideal, libre de los efectos de ruido y decoherencia que caracterizan al *hardware* cuántico real, y sobre un escenario deliberadamente

reducido para garantizar la viabilidad de la comparación. Las limitaciones que se derivan de ambas circunstancias, y que delimitan el camino hacia una eventual aplicación práctica del enfoque, se analizan en la Sección 6.6.

6.6. Limitaciones identificadas

La principal limitación del modelo experimental reside en la simplificación de la dimensión espacial del problema de recuperación de flota. En un problema operativo, la asignación de una aeronave a un vuelo no depende únicamente de la disponibilidad temporal del recurso, sino también de su localización física, de la conectividad entre aeropuertos, de la secuencia de vuelos previamente operada y de los requisitos de posicionamiento al final de la ventana de recuperación. El prototipo desarrollado no incorpora de forma explícita restricciones de continuidad espacial entre origen y destino, ni variables que describan la posición inicial y final de cada aeronave.

Esta simplificación no invalida la prueba de concepto, pero sí delimita claramente su alcance. El resultado obtenido debe interpretarse como la validación de un flujo metodológico QUBO-QAOA aplicado a una abstracción temporal del problema, no como una solución completa y directamente desplegable en un centro de control de operaciones. La extensión natural del modelo consistiría en incorporar restricciones de conectividad espacial, de manera que dos vuelos solo pudieran asignarse consecutivamente a la misma aeronave si el destino del primero coincide con el origen del segundo y si existe tiempo suficiente para completar la rotación correspondiente.

Más allá de esta limitación estructural, que es la principal del modelo desarrollado, la validación de la prueba de concepto presentada en la Sección 6.5 se ha obtenido bajo condiciones que difieren sustancialmente de las que regirían una eventual aplicación práctica del enfoque sobre problemas de escala operativa real. El resto de esta sección identifica y analiza las limitaciones derivadas de estas circunstancias, de naturaleza tecnológica y metodológica, organizadas en cuatro ejes: las limitaciones del *hardware* cuántico actual, las limitaciones de escalabilidad del circuito, la barrera de la simulación clásica y la dependencia de la calidad del muestreo.

6.6.1. Limitaciones del *hardware* cuántico actual

La totalidad de los resultados experimentales de este trabajo se ha obtenido mediante simulación de vector de estado en el entorno **AerSimulator**, que reproduce el comportamiento de un computador cuántico ideal, sin ruido ni decoherencia. Esta elección, justificada por la necesidad de validar el algoritmo en un entorno determinista y reproducible, deja no obstante fuera del análisis el factor que actualmente constituye la principal barrera para la aplicación práctica de los algoritmos cuánticos: el ruido del *hardware*.

Los dispositivos cuánticos disponibles en la actualidad pertenecen a la denominada era NISQ (*Noisy Intermediate-Scale Quantum*), término acuñado por Preskill (2018) para caracterizar una generación de procesadores con un número de qubits del orden de decenas a centenares, pero cuyas operaciones están afectadas por niveles de ruido que limitan severamente la profundidad de los circuitos ejecutables de forma fiable. En estos dispositivos, cada puerta cuántica introduce una probabilidad de error no despreciable, y los qubits pierden su coherencia cuántica en escalas de tiempo finitas. El conjunto de operaciones que pueden ejecutarse antes de que el ruido acumulado degrade irreversiblemente la información constituye lo que Preskill (2018) describe como el presupuesto de error del dispositivo.

En el contexto de este trabajo, esta limitación implica que la fidelidad de la solución obtenida sobre *hardware* real sería inferior a la observada en simulación. Aunque el modelo matemático y la formulación QUBO sean correctos, como ha quedado demostrado en la validación, la ejecución física del circuito introduciría errores que podrían desplazar la distribución de probabilidad muestreada, reduciendo la probabilidad de aparición del óptimo y dificultando su recuperación por parte del post-procesado. Esta brecha entre el comportamiento ideal y el comportamiento real es coherente con las observaciones de Pellow-Jarman et al. (2024), quienes documentan que el rendimiento de QAOA en presencia de ruido depende sensiblemente de la profundidad del *ansatz* y de la elección del optimizador clásico, factores cuyo efecto resulta indetectable en simulación sin ruido.

6.6.2. Limitaciones de escalabilidad del circuito

Una segunda limitación, estrechamente relacionada con la anterior, concierne a la profundidad del circuito tras su adaptación a la topología de un procesador físico. El circuito QAOA transpilado en este trabajo alcanzó una profundidad de 73 capas sobre **AerSimulator**, un valor moderado posibilitado por el hecho de que el simulador no impone restricciones de conectividad entre qubits: cualquier par de qubits puede interactuar directamente, independientemente de su posición.

Los procesadores cuánticos reales, en cambio, presentan una conectividad limitada entre sus qubits físicos. Las arquitecturas actuales de varios fabricantes adoptan topologías dispersas, como la red hexagonal pesada (*heavy-hex lattice*) característica de los procesadores de IBM (Figura 6.2), en las que cada qubit se conecta únicamente con un número reducido de vecinos. Cuando el algoritmo requiere la interacción entre dos qubits que no son físicamente adyacentes, el transpilador debe insertar secuencias de puertas de intercambio (*SWAP gates*) que desplazan la información a través del chip hasta situar la pareja en posiciones contiguas. Esta operación de enrutamiento incrementa sustancialmente el número de puertas del circuito y, en consecuencia, su profundidad efectiva.

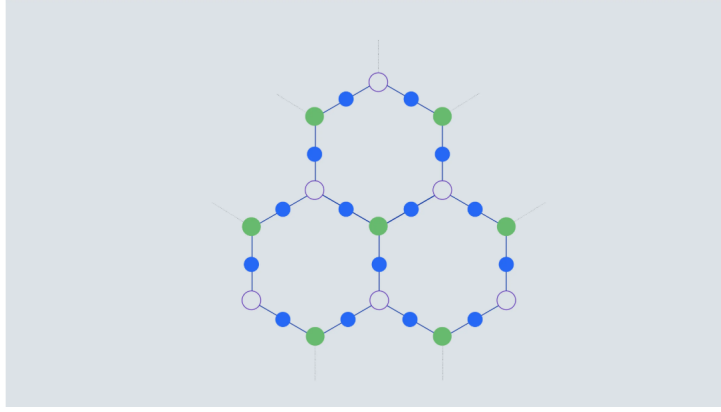


Figura 6.2: Topología *heavy-hex* de los procesadores cuánticos de IBM, en la que cada qubit se conecta únicamente con dos o tres vecinos. Los colores distinguen el papel de cada qubit en el esquema de corrección de errores de IBM (qubits de datos y qubits de medida); a efectos de este trabajo, lo relevante es la conectividad limitada que impone la topología. Fuente: IBM Quantum (2021).

El efecto combinado de ambos fenómenos genera problemas desde el punto de vista de la viabilidad: la conectividad limitada incrementa la profundidad del circuito, y una mayor profundidad agrava la exposición al ruido descrita en el apartado anterior. Para el escenario reducido de este trabajo, este sobre coste sería manejable; sin embargo, para escenarios de escala operativa, con cientos de variables y, por tanto, cientos de qubits, la profundidad del circuito transpilado crecería hasta niveles incompatibles con el presupuesto de error de los dispositivos NISQ actuales.

6.6.3. La barrera de la simulación clásica

Una tercera limitación, de naturaleza distinta a las anteriores, afecta no al *hardware* cuántico sino a la propia herramienta de validación empleada. Como se documentó en la Sección 6.1, el intento de simular el escenario intermedio de aproximadamente 50 variables resultó en un requerimiento de memoria de unos 65 500 terabytes, muy por encima de los recursos disponibles. Esta barrera no es una limitación del algoritmo, sino una propiedad fundamental de la simulación clásica de sistemas cuánticos: el coste de representar el estado de un sistema de n qubits crece como 2^n , lo que hace que la simula-

ción determinista resulte intratable más allá de aproximadamente cuarenta o cincuenta qubits.

Esta limitación tiene una consecuencia metodológica importante. El esquema de validación empleado en este trabajo, basado en la comparación directa de QAOA contra un *solver* clásico exacto, solo es aplicable en el régimen en el que ambos enfoques son ejecutables. Precisamente en el rango de tamaños en el que la computación cuántica comenzaría a resultar interesante (allí donde la simulación clásica deja de ser viable) desaparece la posibilidad de contrastar el resultado cuántico contra una referencia exacta. La validación rigurosa de la corrección, por tanto, está intrínsecamente confinada a la escala reducida, y el salto hacia escalas mayores exige métodos de validación alternativos, como los *benchmarks* contra *solvers* heurísticos de alta calidad o el análisis de cotas teóricas, según se discute en la literatura sobre escalado de aplicaciones cuánticas industriales (Kaushik et al. (2025)).

6.6.4. Dependencia de la calidad del muestreo

La selección de la solución descrita en la Sección 5.7 descansa sobre una premisa: que el óptimo del problema aparezca en el muestreo con probabilidad apreciable, de modo que figure al menos una vez entre las configuraciones obtenidas. Si esta premisa se cumple, la solución se identifica sin dificultad; si no, el óptimo queda fuera del conjunto de configuraciones muestreadas y resulta irrecuperable, con independencia de cómo se procesen los resultados.

En el entorno de simulación ideal empleado en este trabajo, esta premisa se satisface y el óptimo es seleccionado correctamente. Sin embargo, su cumplimiento no está garantizado en condiciones más realistas. En presencia del ruido característico del *hardware* NISQ, o ante una profundidad de *ansatz* insuficiente, la distribución preparada por el circuito puede degradarse hasta asignar al óptimo una probabilidad tan baja que no llegue a aparecer en un número finito de disparos. La fiabilidad del enfoque queda así condicionada por la calidad del estado cuántico preparado, un factor que en este trabajo se ha mantenido controlado gracias a la simulación determinista, pero que

constituiría una dificultad central en una eventual ejecución sobre hardware real.

6.6.5. Síntesis y proyección

En conjunto, las limitaciones identificadas delimitan con precisión el alcance de los resultados de este trabajo: se ha validado la corrección de un enfoque cuántico para la recuperación de flota en un escenario reducido y bajo condiciones ideales de simulación, pero la extrapolación de este resultado a un entorno de aplicación práctica está condicionada por la simplificación espacial del modelo, por el ruido del *hardware* actual, por el crecimiento de la profundidad del circuito en topologías reales, por la imposibilidad de validar mediante simulación clásica más allá de cierta escala, y por la creciente carga del post-procesado clásico. Lejos de invalidar la prueba de concepto, estas limitaciones definen la frontera tecnológica en la que se inscribe el trabajo y trazan las direcciones en las que debería avanzar la investigación para acercar el enfoque a su aplicabilidad real. Estas direcciones, que incluyen las técnicas de mitigación de errores, las estrategias de reducción de profundidad de circuito y la evolución hacia *hardware* tolerante a fallos, se desarrollan en el Capítulo 7.

Capítulo 7. Conclusiones y líneas de investigación futura

7.1. Conclusiones

El presente trabajo demuestra empíricamente la viabilidad metodológica de formular una versión simplificada del problema de recuperación de flota como QUBO, mapearla a un Hamiltoniano de Ising y recuperar el óptimo global del caso reducido mediante un flujo híbrido basado en QAOA y muestreo, validado contra una referencia clásica exacta. Esta afirmación, que constituye el núcleo de la prueba de concepto, ha quedado respaldada por la coincidencia exacta, variable a variable, y no solo en coste objetivo, entre la solución devuelta por el enfoque cuántico y el óptimo identificado por enumeración exhaustiva (Sección 6.5). Sobre esta conclusión central se articulan el resto de los hallazgos del trabajo.

La consecución de este resultado aporta evidencia de la corrección de la cadena metodológica desarrollada para el caso reducido analizado en el Capítulo 4 y en los capítulos de implementación. La formulación del *Aircraft Recovery Problem* como un modelo con restricciones de unicidad y continuidad, su conversión a una formulación QUBO, el mapeo al Hamiltoniano de Ising y la construcción del circuito variacional constituyen un proceso cuya integridad queda confirmada por el propio resultado: cualquier error estructural en alguna de estas etapas, un término de penalización mal construido, un signo equivocado o una restricción omitida, habría desplazado el estado fundamental del Hamiltoniano y habría impedido la recuperación del óptimo. La equivalencia obtenida constituye, en este sentido, una validación de extremo a extremo del flujo metodológico aplicado al caso estudiado.

Más allá de la validación global, el trabajo aporta dos hallazgos metodológi-

cos de interés que matizan y enriquecen la conclusión principal. El primero concierne a la calibración del coeficiente de penalización. La caracterización empírica del espacio de soluciones reveló una densidad de factibilidad extremadamente baja, lo que obligó a situar el coeficiente P por encima del rango teórico derivado de la cota de factibilidad estricta (Sección 5.4). Este resultado, lejos de constituir una desviación del marco teórico, confirma que la elección del coeficiente de penalización en formulaciones QUBO destinadas a QAOA depende no solo de la magnitud de la función objetivo, sino también de la estructura combinatoria del espacio factible.

El segundo hallazgo se refiere a la lectura de la solución a partir del muestreo. Como se documentó en la Sección 6.4, el óptimo global no era el estado más frecuente de la distribución muestreada, que estaba dominada por una solución subóptima de cancelación total. Sin embargo, esto no impidió su recuperación: para identificar el óptimo basta con que el estado preparado por QAOA le asigne una probabilidad apreciable, de modo que aparezca en el muestreo y pueda seleccionarse como la mejor solución factible obtenida, sin necesidad de que sea la más probable. Esta observación pone de manifiesto que la fiabilidad del enfoque depende de la calidad del estado cuántico preparado, un factor controlado en el entorno de simulación ideal de este trabajo pero condicionado por el ruido en una eventual ejecución sobre *hardware* real.

Es imprescindible delimitar con precisión el alcance de estas conclusiones para evitar sobreinterpretaciones. El trabajo no demuestra ventaja cuántica: en el escenario reducido empleado, un *solver* clásico de fuerza bruta resuelve el problema en un tiempo inferior al del enfoque cuántico, como cabe esperar a esta escala. El valor del resultado no reside en una comparación de eficiencia, sino en la demostración de viabilidad y corrección del enfoque como prueba de concepto. Del mismo modo, la validación se ha obtenido en un entorno de simulación ideal, libre de los efectos de ruido y decoherencia que caracterizan al *hardware* real, y sobre un escenario deliberadamente dimensionado para poder realizar la simulación. (Sección 4.5). Las limitaciones que se derivan de estas circunstancias, analizadas en detalle en la Sección 6.6, no invalidan la prueba de concepto: definen la frontera tecnológica en la que se inscribe

el trabajo y trazan las direcciones en las que debe avanzar la investigación para acercar el enfoque a su aplicabilidad real.

7.2. Líneas de investigación futura

Las limitaciones identificadas a lo largo del trabajo no constituyen puntos finales, sino el punto de partida natural para su continuación. La primera y más inmediata línea de investigación consiste en trasladar la ejecución del algoritmo desde el simulador de vector de estado hacia *hardware* cuántico real. Esta transición, que expondría el algoritmo a los niveles de ruido característicos de la era NISQ ((Preskill, 2018)), exige la incorporación de técnicas de mitigación de errores que compensen la degradación de la fidelidad en la preparación del estado. El análisis del comportamiento de QAOA en estas condiciones es especialmente pertinente porque, como han documentado Pellow-Jarman et al. (2024), el rendimiento relativo de los distintos optimizadores clásicos y la elección de la profundidad del *ansatz* adquieren una relevancia que resulta indetectable en simulación sin ruido. Una caracterización sistemática de esta sensibilidad sobre el problema de recuperación de flota constituiría una extensión directa de este trabajo.

Estrechamente ligada a la anterior se encuentra la línea orientada a la reducción de la profundidad del circuito. La ejecución sobre procesadores reales, con topologías de conectividad limitada, incrementa sustancialmente el número de puertas necesarias para enrutar la información entre qubits no adyacentes, agravando a su vez la exposición al ruido. La investigación en estrategias de transpilación adaptadas a la topología del *hardware*, en esquemas de mapeo que minimicen la inserción de puertas de intercambio y en variantes del *ansatz* que reduzcan la profundidad sin sacrificar expresividad, resulta crítica para que el enfoque sea viable en escenarios de tamaño operativo.

Una tercera dirección, de naturaleza metodológica, aborda el problema de la validación más allá de la barrera de la simulación clásica. El esquema de comparación directa contra un *solver* exacto empleado en este trabajo deja de ser aplicable precisamente en el rango de tamaños en el que la computación cuántica comenzaría a resultar interesante, allí donde la simulación clásica determinista se vuelve intratable. El avance hacia escenarios de mayor escala exige, por tanto, el desarrollo de métodos de validación alternativos, ya sea

mediante la comparación contra *solvers* heurísticos clásicos de alta calidad o mediante el análisis de cotas teóricas derivadas de relajaciones del problema, en línea con las prácticas documentadas en la literatura sobre escalado de aplicaciones cuánticas industriales ((Kaushik et al., 2025; Sipos et al., 2025)).

Finalmente, la línea de investigación de mayor alcance conceptual consiste en la progresión desde la recuperación aislada de la flota hacia el problema de recuperación integrada. El presente trabajo ha abordado deliberadamente el Aircraft Recovery Problem de forma aislada, por ser el recurso con mayor rigidez física y el primer eslabón de la cadena de recuperación. Sin embargo, la literatura advierte de que la resolución secuencial e independiente de cada recurso puede conducir a soluciones globalmente subóptimas, ya que una reasignación de flota óptima en términos técnicos puede invalidar los tiempos legales de descanso de las tripulaciones o deteriorar las conexiones de los pasajeros ((Hu et al., 2016; Maher, 2015)). La capacidad de los modelos QUBO para representar interacciones entre variables binarias posiciona a los algoritmos cuánticos como una vía potencial de investigación para avanzar hacia formulaciones más integradas del problema. La extensión del modelo desarrollado para incorporar simultáneamente las restricciones de tripulaciones y pasajeros, aprovechando la escalabilidad estructural de la formulación, representa el horizonte último de esta línea de investigación: un optimizador capaz de generar soluciones de recuperación más integradas y evaluables en plazos compatibles con la urgencia operativa del centro de control de operaciones.

Bibliografía

- Abbas, A., Ambainis, A., Augustino, B., Bärtzchi, A., Buhrman, H., Coffrin, C., Cortiana, G., Dunjko, V., Egger, D. J., Elmegreen, B. G., Franco, N., Fratini, F., Fuller, B., Gacon, J., Gonciulea, C., Gribling, S., Gupta, S., Hadfield, S., Heese, R., ... Zoufal, C. (2024). Challenges and opportunities in quantum optimization. *Nature Reviews Physics*, 6(12), 718-735.
- Clausen, J., Larsen, A., Larsen, J., & Rezanova, N. J. (2009). Disruption management in the airline industry—Concepts, models and methods. *Computers & Operations Research*, 37(5), 809-821.
- Eggenberg, N., Bierlaire, M., & Salani, M. (2007). *A column generation algorithm for disrupted airline schedules* (inf. téc.). École Polytechnique Fédérale de Lausanne. <https://infoscience.epfl.ch/handle/20.500.14299/18732>
- Einstein, A., Podolsky, B., & Rosen, N. (1935). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Physical Review*, 47(10), 777-780.
- Farhi, E., Goldstone, J., & Gutmann, S. (2014, 14 de noviembre). *A Quantum Approximate Optimization Algorithm*. arXiv: 1411.4028. <https://arxiv.org/abs/1411.4028>
- Farhi, E., Goldstone, J., Gutmann, S., & Sipser, M. (2000, 28 de enero). *Quantum Computation by Adiabatic Evolution*. arXiv: quant-ph/0001106. <https://arxiv.org/abs/quant-ph/0001106>
- Glover, F. (1989). Tabu Search—Part I. *INFORMS Journal on Computing*, 1(3), 190-206.
- Glover, F., Kochenberger, G., & Du, Y. (2018, 13 de noviembre). *A Tutorial on Formulating and Using QUBO Models*. arXiv: 1811.11538. <https://arxiv.org/abs/1811.11538>

- Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212-219.
- Hinsenveld. (2024). Proactive Airline Disruption Management: Increasing Schedule Robustness by Simulating Flexibility Measures. <https://resolver.tudelft.nl/uuid:9fef9181-823b-4b25-99fe-fe8062ffbfef>
- Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. The MIT Press.
- Hu, Y., Song, Y., Zhao, K., & Xu, B. (2016). Integrated recovery of aircraft and passengers after airline operation disruption based on a GRASP algorithm. *Transportation Research Part E: Logistics and Transportation Review*, 87, 97-112.
- IBM Quantum. (2021). *The IBM Quantum heavy hex lattice*. Consultado el 17 de junio de 2026, desde <https://www.ibm.com/quantum/blog/heavy-hex-lattice>
- IBM Quantum. (2026). Quantum Approximate Optimization Algorithm Tutorial [Accedido: 19 de junio de 2026].
- Jiang, J., Lin, J., Gu, Y., Lv, Y., & Wu, C.-L. (2025, septiembre). *A Benders and column generation method to the integrated airline schedule and aircraft recovery with gate reassignment*. arXiv: 2509.02404. <https://arxiv.org/abs/2509.02404>
- Kaushik, A., Kim, S. H., Aboumrad, W., Roetteler, M., Topi, A., & Ashworth, R. (2025, abril). *Quantum Computing for Optimizing Aircraft Loading*. arXiv: 2504.01567. <https://arxiv.org/abs/2504.01567>
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598), 671-680.
- Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3), 497-520.
- Lucas, A. (2013, febrero). *Ising formulations of many NP problems*. arXiv: 1302.5843. <https://arxiv.org/abs/1302.5843>
- Maher, S. J. (2015). Solving the Integrated Airline Recovery Problem Using Column-and-Row Generation. *Transportation Science*, 50(1), 216-239.

- Nielsen, M. A., & Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.
- Pellow-Jarman, A., McFarthing, S., Sinayskiy, I., Park, D. K., Pillay, A., & Petruccione, F. (2024). The effect of classical optimizers and Ansatz depth on QAOA performance in noisy devices. *Scientific Reports*, *14*(1), 16011.
- Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. *Quantum*, *2*, 79.
- Roch, C., Ratke, D., Nüßlein, J., Gabor, T., & Feld, S. (2023). The Effect of Penalty Factors of Constrained Hamiltonians on the Eigenspectrum in Quantum Annealing. *ACM Transactions on Quantum Computing*, *4*(2), 1-18.
- Rodrigues, J., Turoboś, F., Lenartowicz, M., Puchała, Z., Klimek, M., Hendzel, K., & Gepner, P. (2025, octubre). *Disruption Management in Airline Operations: A Solver-based Approach using Time-Space Network Optimization*. arXiv: 2510.26831. <https://arxiv.org/abs/2510.26831>
- Rosenberger, J. M., Johnson, E. L., & Nemhauser, G. L. (2003). Rerouting Aircraft for Airline Recovery. *Transportation Science*, *37*(4), 408-421.
- Sipos, D., Czégel, A., & Tóth, B. G. (2025, abril). *Variational Quantum Optimization Benchmark Suite for Airline Crew Pairing and More*. arXiv: 2504.11899. <https://arxiv.org/abs/2504.11899>
- Su, Y., Xie, K., Wang, H., Liang, Z., Chaovalitwongse, W. A., & Pardalos, P. M. (2021). Airline Disruption Management: A Review of Models and Solution Methods. *Engineering*, *7*(4), 435-447.
- Wang, N., Wang, H., Pei, S., & Zhang, B. (2023). A Data-Driven Heuristic Method for Irregular Flight Recovery. *Mathematics*, *11*(11), 2577.
- Zhang, D., Lau, H. C., & Yu, C. (2015). A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems. *Computers & Industrial Engineering*, *87*, 436-453.
- Zhou, L., Wang, S.-T., Choi, S., Pichler, H., & Lukin, M. D. (2020). Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Physical Review X*, *10*(2), 021067.