FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Replanning Flight Schedules Using Quantum Computing

André Mamprin Mori

Mestrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha, PhD

Co-Supervisor: António Castro, PhD

July 24, 2022

Replanning Flight Schedules Using Quantum Computing

André Mamprin Mori

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Sérgio Nunes, PhD
Referee: Rui Pedro Lopes, PhD
Supervisor: Ana Paula Rocha, PhD

July 24, 2022

Abstract

The mission of the airlines' *Operational Control Center* (OCC) is to track the flights' performance by permanently monitoring the existing operational plans. Operational plans result from a previous optimization problem to maximize the airline's revenue by efficiently managing limited and costly resources. Unexpected events inevitably occur during a flight course and may or may not impact the airline's operational plan. The changes in the flight's scheduled plan may have several implications: changes in the operational plan of subsequent flights, scheduling of crew members, travel plans of passengers, and the airplane's maintenance plan. The process to recover the schedule after disruptions is known as *Airline Disruption Management* and is run by the airline's OCC.

This work focuses only on the airplane component, known as *Aircraft Recovery Problem* (ARP), which involves decisions concerning aircraft to flight assignments in situations where unforeseen events have disrupted the existing flight schedule. A system to solve the ARP used in real-time by airlines' OCC should provide quick and reliable solutions. One of the main difficulties of solving the ARP is the overall complexity since a single disruption may cause a chain effect that can disrupt many other subsequent flights. Computing the total solution space is not feasible for today's classical computers due to the number of possible solutions and constraints while trying to minimize the impact of the disruption.

Quantum computing (QC) is an emerging field that applies the principles of quantum mechanics to computers. Quantum computers are still very limited in terms of memory but have been proven efficient in solving complex problems currently not feasible for classical computers. The QC field has been growing research interest in the past few years, with many applications in optimization problems. The Quadratic Unconstrained Binary Optimization (QUBO) model has been one of the primary options for solving optimization problems using QC, with different companies such as IBM, D-wave and Microsoft building quantum computers dedicated to solving it.

In this study, the ARP was modeled as a QUBO and was solved by classical and hybrid solvers, comparing the final operational cost and resulting flight plan. Actual data from TAP Air Portugal, Portugal's leading airline, was used to analyze the performance of the implementation in a real-case scenario. The implementation was compared to another previous classical implementation of an ARP solution in terms of execution time and the resulting operational plan. Due to current limitations in terms of memory limitation and availability of quantum computers, a hybrid solver which combines classical and quantum computing had to be used. Still, this study concluded that modeling the ARP as a QUBO proved efficient and feasible. Solving the QUBO with hybrid computers provided a quick, feasible, low-cost solution and obtained similar results to the compared previous implementation.

Keywords: Flight Rescheduling, Quantum Computing, Aircraft Recovery Problem

Resumo

A missão do *Operational Control Center* (OCC) das companhias aéreas é acompanhar o desempenho dos voos através da monitorização permanente dos planos operacionais existentes. Os planos operacionais resultam de um problema de optimização anterior para maximizar as receitas da companhia aérea, gerindo eficientemente recursos limitados e dispendiosos. Eventos inesperados ocorrem inevitavelmente durante um curso de voo e podem ou não ter impacto no plano operacional da companhia aérea. As alterações no plano programado do voo podem ter várias implicações: alterações no plano operacional dos voos subsequentes, programação dos membros da tripulação, planos de viagem dos passageiros e o plano de manutenção do avião. O processo de recuperação do plano após perturbações é conhecido como *Airline Disruption Management* e é gerido pelo OCC da companhia aérea.

Este trabalho centra-se apenas na componente de aviões, conhecida como *Aircraft Recovery Problem* (ARP), que envolve decisões relativas à atribuição de voos a aeronaves em situações de perturbações ao horário de voo existente. Um sistema para resolver o ARP utilizado em tempo real pelo OCC das companhias aéreas deve fornecer soluções rápidas e fiáveis. Uma das principais dificuldades para resolver o ARP é a complexidade global, uma vez que uma única perturbação pode causar um efeito em cadeia e perturbar voos subsequentes. A computação do espaço total de solução não é viável para os computadores clássicos actuais devido ao número de soluções e constrangimentos possíveis enquanto se tenta minimizar o impacto da perturbação.

A computação quântica (QC) é um campo emergente que aplica os princípios da mecânica quântica aos computadores. Os computadores quânticos ainda são muito limitados em termos de memória, mas provaram ser eficientes na resolução de problemas complexos que actualmente não são viáveis para computadores clássicos. O campo da QC tem vindo a aumentar o interesse da investigação nos últimos anos, com muitas aplicações em problemas de optimização. O modelo Quadratic Unconstrained Binary Optimization (QUBO) tem sido uma das principais opções para resolver problemas de optimização utilizando QC, com diferentes empresas como a IBM, D-wave e Microsoft a construírem computadores quânticos dedicados à sua resolução.

Neste estudo, o ARP foi modelado como um QUBO e resolvido por *solvers* clássicos e híbridos, comparando o custo operacional e o plano de voo resultante. Dados reais da TAP Air Portugal, a principal companhia aérea portuguesa, foram utilizados para analisar o desempenho do modelo num cenário real. A implementação foi comparada a uma solução clássica do ARP em termos de tempo de execução e do plano operacional resultante. Devido às limitações de memória e disponibilidade de computadores quânticos, utilizou-se um *solver* híbrido que combina computação clássica e quântica. Este estudo concluiu que a modelação do ARP como um QUBO provou ser eficiente e viável. A resolução do QUBO com *solvers* híbridos forneceu uma solução rápida, viável e de baixo custo obtendo resultados semelhantes aos da implementação anterior.

Keywords: Replaneamento de Voo, Computação Quântica, Aircraft Recovery Problem

Acknowledgements

I am immensely grateful to my supervisors, Ana Paula Rocha and António Castro, for all the help and guidance throughout the development of this dissertation. I am also extremely thankful to my family for all the support given during these years living and studying abroad, especially during the hard times. Finally, I would like to thank all my friends I made along the way for their friendship and the incredible moments we spent together.

André

"When everything seems to be going against you,
remember that the airplane takes off against the wind,
not with it."

Henry Ford

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation and Objectives . . . 2
- 1.3 Main Contributions . . . 2
- 1.4 Document Structure . . . 2

2 State of the Art **5**
- 2.1 Background . . . 5
  - 2.1.1 Aircraft Scheduling Process . . . 5
  - 2.1.2 Quantum Computing . . . 6
  - 2.1.3 Quantum Platforms . . . 9
  - 2.1.4 Quadratic Unconstrained Binary Optimization . . . 10
- 2.2 Literature Review . . . 12
  - 2.2.1 Aircraft Recovery Problem . . . 12
  - 2.2.2 Quantum Algorithms . . . 16
- 2.3 Summary . . . 18

3 Aircraft Recovery Problem **19**
- 3.1 Problem Description . . . 19
  - 3.1.1 Operational Plan . . . 20
  - 3.1.2 Events and Disruptions . . . 20
  - 3.1.3 Constraints . . . 20
  - 3.1.4 Costs . . . 21
  - 3.1.5 Optimization Criteria . . . 21
- 3.2 Dataset Analysis . . . 22
  - 3.2.1 Aircraft Analysis . . . 22
  - 3.2.2 Flights Analysis . . . 23
  - 3.2.3 Events Analysis . . . 25
- 3.3 Summary . . . 29

4 Quantum Approach to Flight Rescheduling **31**
- 4.1 Two Stage Algorithm . . . 31
  - 4.1.1 Initialization . . . 31
  - 4.1.2 First Stage . . . 32
  - 4.1.3 Second Stage . . . 32
- 4.2 Aircraft Feasible Flights Graph . . . 33
  - 4.2.1 Graph Structure . . . 33
  - 4.2.2 Generation Process . . . 34

4.3 Quadratic Unconstrained Binary Optimization Modeling . . . 36
4.3.1 Penalties . . . 37
4.3.2 Objective Function . . . 38
4.3.3 Quadratic Matrix . . . 39
4.4 Summary . . . 39

5 Experiments and Results **41**
5.1 Experiments Description . . . 41
5.1.1 First Experiment Description . . . 41
5.1.2 Second Experiment Description . . . 42
5.2 Metrics . . . 43
5.2.1 First Experiment Metrics . . . 43
5.2.2 Second Experiment Metrics . . . 43
5.3 Result Analysis . . . 43
5.3.1 First Experiment Result Analysis . . . 44
5.3.2 Second Experiment Result Analysis . . . 46
5.4 Summary . . . 47

6 Conclusions and Future Work **49**
6.1 Conclusions . . . 49
6.2 Main Difficulties . . . 50
6.3 Main Contributions . . . 51
6.4 Future Work . . . 51

References **53**

A Experiments Execution Times **57**
A.1 First Experiment Results . . . 57
A.2 Second Experiment Results . . . 57
A.2.1 CPLEX Results . . . 57
A.2.2 D-wave's LeapHybridSolver Results . . . 57

List of Figures

2.1 Simplified overview of the airline scheduling process 6
2.2 Bloch sphere representation. Adapted from [26] 7

3.1 Cost distribution through different flight stages 21
3.2 Aircraft fleet distribution percentage 23
3.3 Aircraft model passenger capacity 24
3.4 Flights date distribution 26
3.5 Flights distribution 26
3.6 Top 5 main departure airports 27
3.7 Event causes and hard changes distribution 28
3.8 Event date distribution 28

4.1 Proposed algorithm overview 32
4.2 Initialization stage overview 32
4.3 Generated graph for an Airbus A319 34
4.4 Generated graph for an Airbus A319, including edge keys and the case of more than one edge between two nodes 35
4.5 Partial representation of a quadratic matrix 39

5.1 D-wave's operation and timing process. Available at [31] 44
5.2 Execution times of the first experiment. 45
5.3 Execution times of the second experiment 46

List of Tables

2.1 Some classical constraints and their equivalent quadratic penalties. 11
2.2 Comparison of ARP previous works 15

3.1 Original dataset summary 22
3.2 Aggregate and reduced dataset summary 23
3.3 Aircraft table description 24
3.4 Aircraft models and fleet types 24
3.5 Flights table description 25
3.6 Events table description 26

5.1 First experiment disruptions details 42
5.2 Second experiment disruptions details 42
5.3 Delay and swap penalties values used in the program 44
5.4 Results of the first experiment. 45
5.5 Results of the second experiment 48

A.1 First experiment results 58
A.2 Second experiment results for IBM's CPLEX regarding disruptions on flights 712, 686 and 152 59
A.3 Second experiment results for IBM's CPLEX regarding disruptions on flights 129 and 916 60
A.4 Second experiment results for D-wave's LeapHybridSolver regarding disruptions on flights 712, 686 and 152 61
A.5 Second experiment results for D-wave's LeapHybridSolver regarding disruptions for flights 129 and 916 62

Abbreviations

ABC	Artificial Bee Colony
ACI	Airports Council International
ACO	Ant Colony Optimization
ARP	Aircraft Recovery Problem
ATC	Air Traffic Controller
CEA	Classical Evolution Algorithm
ICAO	International Civil Aviation Organization
LNS	Large Neighbourhood Search
LOF	Line of Flight
NB	Narrow-body
OCC	Operations Control Center
PSO	Particle Swarm Optimization
QA	Quantum Annealing
QABC	Quantum-inspired Artificial Bee Colony
QACO	Quantum Ant Colony Optimization
QAOA	Quantum Approximate Optimization Algorithm
QC	Quantum Computing
QDK	Quantum Development Kit
QEA	Quantum-inspired Evolutionary Algorithm
QPU	Quantum Processing Unit
QSEA	Quantum-inspired Swarm Evolution Algorithm
Qubit	Quantum bit
QUBO	Quadratic Unconstrained Binary Optimization
SA	Simulated Annealing
SDK	Software Development Kit
TS	Tabu Search
VQE	Variational Quantum Eigensolver
WB	Wide-body

Chapter 1

Introduction

This chapter introduces the the goal of this dissertation. Section 1.1 starts by providing the context of this dissertation. Section1.2 provides the motivation and the objectives to be accomplished in this work. Section 1.3 presents the main contributions expected to be achieved by the end of the dissertation. Finally, section 1.4 presents the structure of this document.

1.1 Context

The airline industry encounters disruptions in its flight schedules daily. The leading causes of these disruptions are usually weather conditions, delays, unexpected maintenance and airport closure. The impacts of these issues may be low/non-existent, such as a minor delay, but it can also start a chain of events that affect many more subsequent flights which increases the overall financial losses.

In 2020, EUROCONTROL released a report regarding flight delays and cancellations in Europe during the year of 2019 [14]. The report concluded that the arrival punctuality in Europe during that year was 77.6%, which translates to an average of 13.1 minutes delay for each flight. The report also stated that 43.5% of the total delay was *reactionary delays* caused by previous flights' delays. This data shows that disruptions are a common problem faced by airlines, with almost 25% percent of flights not arriving punctually. The data also shows that nearly 45% of the delays is caused by a chain of events which propagates throughout subsequent flights.

Recovering the schedule after disruptions is known as disruption management, which consists of recovering the disturbance associated with aircraft, crew, and passengers, generally delay and cost. This work focuses only on the aircraft component, known as the *Aircraft Recovery Problem* (ARP). The goal of the ARP is to perform a rescheduling of flights while trying to minimize the total costs and delays. The main challenge when solving the ARP is that solutions must be generated quickly since many disruptions occur during the flight plan [32] while the problem is very complex and demanding.

Quantum computing (QC) is a field that has gained much interest in recent years. Quantum computers can solve problems that are currently not feasible in a classical computer by taking advantage of quantum mechanics and quantum parallelism. Due to its benefits, the QC field may be able to efficiently solve the ARP providing a fast and low cost solution.

1.2 Motivation and Objectives

The Aircraft Recovery Problem (ARP) is a highly complex optimization problem that requires a quick and feasible solution while minimizing operational costs. Airlines face this problem almost daily, with some cases having a more significant impact than others. Disruptions may lead to a chain of events affecting subsequent flights, resulting in a more severe financial loss for the company. Especially in recent years, the companies have been more pressured into avoiding financial impacts due to the global pandemic that grounded many airlines due to travel restrictions and reduced demands.

Quantum Computing (QC) has been gaining interest in the research field in finding applications for problems that are not currently feasible for classical computers. Applying QC in optimization problems has been one of the main fields of study in recent years especially using the Quadratic Unconstrained Binary Optimization (QUBO) model, which companies such as IBM, D-wave and Microsoft have built quantum computers designed for solving this type of problems.

With the development of this dissertation, the impact of using QC in solving the ARP will be assessed. The main goal of this work is to develop a quantum approach capable of solving the problem and comparing its performance, such as execution time and resulting operational plan, with other previously implemented classical approaches.

1.3 Main Contributions

The main contributions expected to be achieved by the end of this dissertation are:

- Developing and implementing a model for the Aircraft Recovery Problem (ARP)
- Solve the model using quantum and classical solvers
- Analyzing and assessing the impact of quantum computing when solving the ARP compared to solving using classical solvers

1.4 Document Structure

The remainder of the document's division is as follows. Chapter 2 introduces background information on the main topics of the dissertation, such as quantum computing, aircraft scheduling process, introducing the ARP, and quantum algorithms and also revises the existing literature and previous works on quantum computing and the aircraft recovery problem. The problem statement

is presented in chapter 3 along with an analysis of the dataset used in this work. The proposed solution is detailed in chapter 4. Chapter 5 presents the experiments performed to evaluate the implementation performance. Lastly, chapter 6 presents the work's conclusion, the main difficulties and contributions, and suggests ideas to be used in future work. Appendix A contains the execution times and solutions of the experiments conducted in this work.

Chapter 2

State of the Art

2.1 Background

This section provides background regarding airline scheduling and introduces the Aircraft Recovery Problem (ARP) and the Quantum Computing (QC) concept. Section 2.1.1 explains the scheduling process performed by airlines, including dealing with disruptions such as the ARP and its main challenges. Section 2.1.2 provides a description of QC, the correlation with quantum mechanics and its applications in developing quantum algorithms. Section 2.1.3 presents different quantum platforms available for usage. Section 2.1.4 gives an overview of the quadratic unconstrained binary optimization (QUBO) model, which was used to solve the problem.

2.1.1 Aircraft Scheduling Process

The airline scheduling process is a continuous activity consisting of long and short-term actions to utilize all of the company's resources efficiently and generate an operational plan. Based on Kohl et al. [22], the scheduling process splits the aircraft, crew and passengers into separate entities so that their scheduling and handling can happen independently. Figure 2.1 illustrates a simplified overview of the airline scheduling process.

The first step into the scheduling plan is route planning, which happens long before operating. This step optimally schedules all the airline's flight routes throughout a specified period. The resulting timetable includes all the flights that will be performed, including their date and time, usually tending to maximize the company's revenue.

The second step is scheduling the company's resources into the previously generated flight plan. It assigns the crew, aircraft fleet type, and each individual aircraft to the flight plan. This step is also crucial to the airline since an optimal schedule may maximize the company's revenue.

The last step is called disruption management, which is a replanning activity that comes after the initial planning phase and may be executed during the operation in question [9], and occurs during the day of operations. Disruption management deals with three primary resources affected by the disruptions in the airline industry: the aircraft, crew, and passengers. Traditionally, disruptions are solved sequentially: first, the airline's operational control center OCC deals with

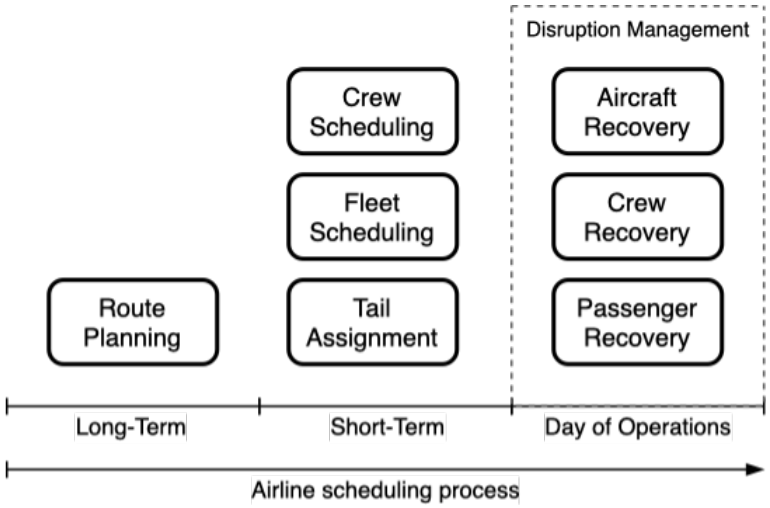


Figure 2.1: Simplified overview of the airline scheduling process

rescheduling the aircraft, then the crew is reallocated to the new flight plan, and finally, the passengers' itineraries are dealt with [8]. The problem of recovering the aircraft schedule is known as the *Aircraft Recovery Problem* (ARP).

The ARP consists of delaying, canceling or swapping flights to recover the flight plan keeping the changes, delays and costs as low as possible [29]. This problem is recurrent, and airlines face it daily, requiring a fast and feasible solution while keeping the financial losses as low as possible. Chapter 3 further defines the ARP as well as introduces important concepts considered in this work.

2.1.2 Quantum Computing

Quantum Computing (QC) is the combination of quantum mechanics with computing. It offers more computational power than classical computers and can solve problems that are not currently feasible by using quantum principles. Further details about QC will be presented in this section, including its main characteristics, applications and algorithms.

2.1.2.1 Quantum Bit

The unit of quantum information in QC is known as a quantum bit (qubit) [30]. The qubit is a unit vector in a two-dimensional complex vector space, and its standard basis representation can be written as $\{|0\rangle, |1\rangle\}$ [27]. The basis states $|0\rangle$ and $|1\rangle$ correspond to the classical bits 0 and 1 states, respectively. The states' basis representation derives from a notation known as *bracket* proposed by Dirac [11] to represent a Hilbert-space vector in quantum mechanics. Equivalently, basis states could also be written in matrix form such as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.1}$$

2.1.2.2 Quantum Superposition

Classical bits can be either in state 0 or state 1, while qubits can be in states other than $|0\rangle$ or $|1\rangle$. This circumstance is a quantum principle known as *superposition*, which allows a qubit to be on multiple states at once and can be represented as a linear combination of states [25]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \alpha, \beta \in \mathbb{C} \tag{2.2}$$

where values α and β are complex numbers which correspond to the probability of a qubit $|\psi\rangle$ being on state $|0\rangle$ and $|1\rangle$, respectively, such that $|\alpha|^2 + |\beta|^2 = 1$. Due to the fact that the condition $|\alpha|^2 + |\beta|^2 = 1$ applies, equation 2.2 can be written as [26]:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle, \theta, \phi \in \mathbb{R} \tag{2.3}$$

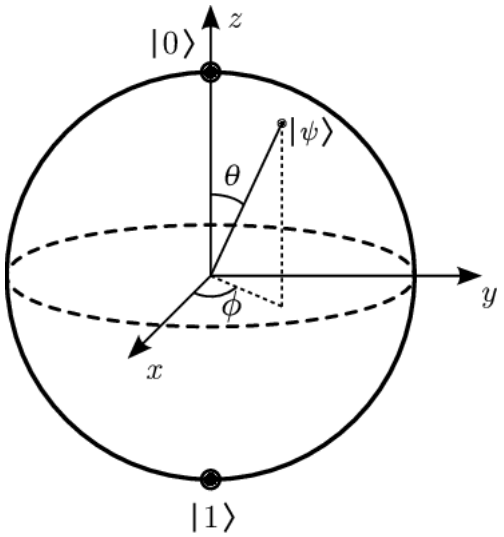


Figure 2.2: Bloch sphere representation. Adapted from [26]

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. This equation visualizes the qubit as a unit vector on the Bloch sphere and the unitary operations as rotations on the sphere, as illustrated in figure 2.2. Each point on the Bloch sphere represents a state, and the rotations correspond to the transformations of the state.

2.1.2.3 Multiple Qubits System

In a system with n-qubits, there are 2^n possible states. If the system has two bits in a classical computer, the possible states would be 00, 01, 10 and 11. In a two-qubit system, there would be four possible states: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. In a multiple qubits system, the qubits can also be in a superposition of states, and the linear combination of states in a two-qubit system can be represented as:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \alpha, \beta, \gamma, \delta \in \mathbb{C} \tag{2.4}$$

where values $\alpha, \beta, \gamma, \delta$ are complex numbers such that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$.

Individual states of n particles combine classically with the Cartesian product, while quantum states combine through the tensor product [27]. The basis state of a two-qubits system might be written, using the tensor product, as $\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$ or more commonly as $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

2.1.2.4 Measurement

In a classical computer, it is possible to obtain a bit's state at any given moment, but when dealing with qubits, it is not as simple since it may be in superposition and needs to be measured. Based on equation 2.2, qubit $|\psi\rangle$ may be measured as $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. In a multiple qubit system, an n-qubit measure can be treated as measuring every single qubit individually [27].

Superposition is basis-dependant [28], meaning that a qubit can be in superposition with respect to basis $\{|0\rangle, |1\rangle\}$, but not to $\{|\uparrow\rangle, |\rightarrow\rangle\}$. Consequently, measuring a qubit on a specific basis can give a deterministic result while giving an unexpected result on another basis.

2.1.2.5 Quantum Entanglement

In a system, if the measurement of a particle affects the measurement of another, this means they are *entangled* [27]. An example of entanglement can be demonstrated as:

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.5}$$

The state presented in equation 2.5 is known as a *Bell State* [28]. It is considered entangled since the first qubit's measurement affects the second and vice-versa. For instance, if no qubit was measured previously, the probabilities of the first qubit being $|0\rangle$ or $|1\rangle$ are both 50%. In another scenario, if the second qubit was measured as 0, the probabilities would change to 100% and 0%, respectively.

2.1.2.6 Quantum Gates

In a classical computer, logic gates accept inputs consisting of bits to perform logical operations and generate an output, which is also consisting of bits. Some examples of these logic gates are: AND, OR, XOR and NOT. Quantum computers also have logic gates known as *quantum gates*. What differs between classical and quantum gates is that the input and output of quantum gates may be arbitrary states [10]. Unlike most classical gates, quantum gates are reversible, making it possible to determine input values from the operation output [3].

Quantum gates are represented by matrices of 2^k dimension, where k represents the number of input/output qubits. Usually, quantum gates tend to act on one or two qubits. The application of quantum gates usually occurs on one or two qubits, and some examples will be demonstrated below.

Identity Gate

The identity gate is defined as an identity matrix, and it does not alter the quantum state when applied. It is applied to one-qubit operations, and its matrix representation is as follows:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.6}$$

Pauli Gates

There are three main Pauli gates: X, Y and Z. They are single-qubit gates and represent the transformation of the state on the X, Y and Z-axis, respectively. Each of these gates, when applied, rotates the state on the respective axis of the Bloch sphere by π radians. The Pauli gates matrices are represented as:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.7}$$

Controlled NOT Gate

The controlled not gate acts on two qubits. It applies the NOT operator on the second qubit if the first qubit is $|1\rangle$. The matrix representation of this gate is as follows:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.8}$$

2.1.3 Quantum Platforms

Quantum computing is still very limited today. The hardware needed is costly and unreliable for real-life applications due to noise introduced in gate operations, resulting in few options available in the market. This section will present and analyze different considered platforms to solve the ARP in this dissertation, presenting the main characteristics, advantages and disadvantages.

IBM Quantum

IBM offers access to different quantum hardware, simulators and programming tools through their IBM Cloud service. There are over 20 systems available to choose from, ranging from one up to 127 qubits, with seven options of *open* access while the others are restricted to IBM Quantum members. IBM has also developed Qiskit [2], an SDK that allows the user to work with quantum computers.

Qiskit has a circuit library that includes quantum gates and a set of pre-built circuits to make it user-friendly for beginners and advanced programmers. It also offers a transpiler that converts

Qiskit's code into an optimized circuit so that users can program for any quantum processor or processor architecture. Finally, it allows users to run their programs on IBM's systems.

Microsoft Quantum

Microsoft is also building a scalable quantum system. Microsoft offers different providers of quantum computers to execute the program on quantum hardware, supporting up to 29 qubits. They have an open cloud ecosystem for quantum computing called Azure Quantum, and it is possible to code using this platform with a language called Q#, which is included in their Microsoft Quantum Development Kit (QDK).

The QDK offers ready-to-use libraries that include standard usage, such as common quantum patterns, and domain-specific, such as chemistry or machine learning. Quantum simulators are available to run a small program instance without actual quantum hardware, and noise simulators are also available to allow the execution of programs to be more realistic and observe its effects. Another feature of the QDK is integrating classical and quantum computing using either Python or .NET libraries.

Quantum Computing Playground

Quantum Computing Playground is a browser-based WebGL Chrome Experiment. It is integrated into a GPU-accelerated quantum computer with a simple IDE and its scripting language, with different quantum gates integrated. The platform offers up to 22 qubits, and it allows running typical quantum algorithms such as Grover's and Shor's.

D-Wave Quantum

D-Wave provides a quantum solution called Leap, a quantum cloud service that leverages classical and quantum resources allowing a hybrid programming experience. Leap provides a browser-based IDE with Ocean, an SDK developed by D-Wave that allows development in its platform. The company's focus is solving problems using quantum annealing, the only solver option available in their platform.

There are two quantum computers available to use: *2000Q* and *Advantage*. They offer over 2000 and 5000 qubits, respectively, but only allow 1 minute of execution per month on the free plan. They also offer three different hybrid solvers that allow 20 minutes on the free plan.

2.1.4 Quadratic Unconstrained Binary Optimization

The Quadratic Unconstrained Binary Optimization (QUBO) model is a mathematical formulation used to solve quadratic combinatorial optimization problems. The model consists of a vector of binary variables representing "yes or no" choices, which multiply values or penalty factors. This value will be added to the final objective function when the variable is set to true.

The QUBO model has also been widely used in quantum computing to solve many different optimization problems, which will be presented during the literature review in section 2.2. Companies such as D-Wave, IBM and Microsoft have built quantum computers dedicated to solving these problems with different algorithms. Classical implementations such as IBM's CPLEX are widely used for solving QUBO problems.

2.1.4.1 Formulation

A QUBO model can be defined with x, a vector of binary decision variables, and Q, a n-by-n square matrix of constants, to minimize or maximize the function f, as presented in equation 2.9. The function is split into two different sums since the model consists of binary variables and the condition $x_{i,j}^2 = x_{i,j}$ is always true. The value $Q_{i,i}$ corresponds to the linear constant and $Q_{i,j}$ corresponds to the quadratic constant.

$$f(x) = \sum_{i=1}^{n} Q_{i,i}x_{i,i} + \sum_{i<j}^{n} Q_{i,j}x_{i,j}, \quad x \in \{0,1\} \tag{2.9}$$

Another possible and more compact formulation of the model can be represented as shown in equation 2.10 [17], where x is the vector of binary decision variables and Q is the n-by-n square matrix of constants.

$$max/min : y = x^t Q x, \quad x \in \{0,1\}^n \tag{2.10}$$

The quadratic matrix Q can be represented in a symmetric form or another form known as *upper triangular*, which is commonly used in the programming libraries, such as IBM's Qiskit. To obtain the matrix in a upper triangular form from a symmetric matrix, for each i and j satisfying $j > i$, add $Q_{j,i}$ to $Q_{i,j}$. The next step is to replace each $Q_{i,j}$, where $j < i$ with the value 0.

2.1.4.2 Penalties

As presented previously in this section, a QUBO model is unconstrained. Still, via reformulation, any constrained problem can be cast into a QUBO by simply including quadratic penalties in the objective function [21]. Table 2.1 presents some known classical constraints and their quadratic penalty equivalencies, which can be used in the objective function. These conversions are used and presented further in section 4.3.1, where the penalty factors for modeling the QUBO are illustrated.

Classical Constraints	QUBO Penalties
$x_1 + x_2 \leq 1$	$P(x_1x_2)$
$x_1 + x_2 \geq 1$	$P(1 - x_1 - x_2 + x_1x_2)$
$x_1 + x_2 = 1$	$P(1 - x_1 - x_2 + 2x_1x_2)$
$x_1 \leq x_2$	$P(x_1 - x_1x_2)$

Table 2.1: Some classical constraints and their equivalent quadratic penalties.

2.2 Literature Review

This section presents existing literature regarding the Aircraft Recovery Problem (ARP) and Quantum Computing (QC). Section 2.2.1 presents and analyzes current ARP implementations, including different approaches and algorithms. Section 2.2.2 illustrates different quantum algorithms implementations and also the usage of Quadratic Unconstrained Binary Optimization (QUBO) modeling.

2.2.1 Aircraft Recovery Problem

This section will present and analyze different proposed algorithms to solve the ARP. The main objective is to demonstrate the different approaches from each author to tackle the problem and compare the obtained results. Table 2.2 presents a summary and comparison of the solutions presented in this section.

Evolutionary Computation methods applied to Operational Control Centers

Ferreira [16] implemented two different algorithms, Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), to solve the ARP. These algorithms were implemented in a Multi-Agent System called MASDIMA (Multi-Agent System for Disruption Management) [7]. The study's goal was to compare the algorithms' performance against two other algorithms, Simulated Annealing (SA) and Hill Climbing (HC), which were already implemented in MASDIMA and successfully solved the ARP.

PSO begins by launching each particle, and for each iteration of the algorithm, the updated particle's fitness value is calculated per the set objective function. The update involves continuously replacing the current aircraft selected for the flight with a different aircraft from the list of acceptable resources, as all of the resources on the list are interchangeable.

Using ants to move across the colony and leave pheromone trails that are proportionate to the quality of the solution the ant provides, the ACO process for generating solutions involves starting the pheromones trail and then keeping it updated through the generation of new solutions. Pheromone trails become more reliable and offer better solutions as time passes, creating more iterations.

The performance of the implemented algorithms was tested by solving five different disruptions and measuring the execution time and solution cost. All five disruptions were solved successfully, resulting in a new flight plan without delay for every algorithm. In terms of execution time, PSO was the second-fastest, behind SA, and ACO was the slowest. PSO provided the second lowest cost, behind HC, and ACO provided the higher cost of all the algorithms.

Ant Colony Optimization

Zegordi et al. [34] developed an Ant Colony Optimization (ACO) algorithm to solve the ARP while also considering the passengers as part of the total cost.

In this solution, the ARP is modeled by a graph with two distinct sets of nodes, one representing aircraft and the other flights and rotations. The role of the ants is as aircraft and must select the flights and rotations while minimizing the total cost, delay and number of cancellations. Canceled flights are represented by nodes that any ant did not visit.

The experiments were conducted using a real-life dataset in two different scenarios, the first being when an aircraft is unavailable for some time and the second being having flight delays. The ACO's performance was tested against a Simulated Annealing (SA) and a Tabu Search (TS) algorithm. The results demonstrated that the algorithm was able to provide the best known solution except for one occasion, while it outperformed the TS algorithm and had a similar performance as the SA.

Two-Stage Heuristic Algorithm

Zhang [35] proposed a Two-Stage Heuristic algorithm to solve the ARP. The primary variable in this model is the *Line of Flight* (LOF). LOFs are a sequence of feasible flights that can be flown by an aircraft on a given day. When generating LOFs, some constraints must be considered: the arrival destination from a flight must be identical to the departure of the next flight, and the time between both flights must be within a defined interval $[T_{min}, T_{max}]$, where T_{min} might be a negative value to allow delays on the departure. After LOFs are generated, an Aircraft Recovery Network is constructed by assigning each LOF to every available aircraft. The only constraint in this phase is that if there is a maintenance plan in a LOF, it must be assigned only to the aircraft assigned to the maintenance. With the network, it generates an Aircraft Recovery Model with the objective function to minimize costs and the required constraints.

Since the number of LOFs increases exponentially with more flights and aircraft, there is a need to reduce the amount of LOFs to provide a solution quickly. The solution proposed by the author is a Two-Stage Heuristic selection for LOFs. The first stage of the algorithm consists of scoring LOFs based on the least amount of swaps and the efficient use of each aircraft, meaning that each aircraft should operate as many flights as possible throughout the day. The LOFs with the lowest scores are excluded after the first stage, and the remainder LOFs are used as input during the next stage. Despite having a reduced number of LOFs, the number of variables is still significant.

The second stage of the algorithm aggregates flow constraints introduced during the recovery model formulation since they were created for each aircraft and airport. A new Aggregate Aircraft Recovery Model is generated with an objective function and fewer constraints. An algorithm was developed first to solve the recovery model and select the "best" LOFs, followed by a Linear Programming solution to obtain the reduced cost for each LOF, and finally set a threshold and select LOFs with a lower cost.

The algorithm proposed outperformed solutions that are used by airlines by minimizing the number of delays, cancellations and costs. The two-stage heuristic proved efficient and managed to reduce the number of LOFs by a significant margin, in a specific case from over 8 million to less than 500.

Large Neighbourhood Search Heuristic Algorithm

Bisaillon et al. [4] proposed a Large Neighbourhood Search (LNS) heuristic approach to solve the ARP. The proposed method is divided into three phases: construction, repair and improvement. This algorithm's goal is to minimize operating, passenger and inconvenience costs.

The first phase starts by randomly sorting the aircraft set to be different every time this step is executed. The following step consists of constructing feasible rotations for each aircraft, starting with the original schedule. When the original rotation is not feasible anymore, flights are delayed to try and find a new feasible schedule. If a solution is not found to solve the disruption, it remains unfixed and addressed in the next phase.

The second phase is the repair phase. It consists of fixing the flight schedule where it remained unfixed by altering other flights' schedules and finding a feasible solution.

The last phase consists of improving the solution generated by the other phases. The improvement consists of slightly delaying flights to try and accommodate additional passengers. The phase stops when there are no possible changes to improve the final cost.

The proposed solution was tested using a dataset obtained from a competition regarding solving the ARP, where the author placed first. The algorithm successfully solves the problem since the approach is insensitive to any dataset instances and is also very fast when providing solutions, being able to use in a real-life scenario.

Reference	Algorithm	Dataset	Optimization	Constraints	Key points
Ferreira [16]	Ant Colony Optimization & Particle Swarm Optimization	Real-scenario	Minimizes total cost, delay and cancellations	1. Swapped aircraft must be from the same fleet. 2. Passengers	Used the same dataset available in this work. Algorithms implemented in a already used system.
Zegordi et al. [34]	Ant Colony Optimization	Real-scenario	Minimizes cost, delay and cancellations	1. Flight either assigned or cancelled. 2. Maintenance. 3. Duty limit. 4. Passengers.	Passengers are included in the recovery.
Zhang [35]	Two-stage Heuristic	Real-scenario	Minimizes cost, delay and cancellations	1. Flight either assigned or cancelled. 2. Maintenance.	Reduces the solution space drastically. Outperforms current airlines' solutions
Bisaillon et al. [4]	Large Neighbourhood Search Heuristic	Unknown	Minimizes operating, passenger and inconvenience costs	1. Swapped aircraft must be from the same fleet. 2. Maintenance. 3. Passengers	The algorithm provides solutions quickly and is not affected by the dataset's instance.

Table 2.2: Comparison of ARP previous works

2.2.2 Quantum Algorithms

In this section, different quantum algorithms will be presented. Some of them are quantum approaches to classical algorithms presented in the section above. The goal is to provide already implemented options and analyze their performance to be considered for the final proposed algorithm. By the end of the section, solutions modeled as a QUBO to solve optimization problems will also be presented.

Quantum-inspired Artificial Bee Colony

Bouaziz et al. [5] proposed a hybridization between the Artificial Bee Colony (ABC) algorithm with QC called Quantum-inspired Artificial Bee Colony (QABC). The authors' main goal was to propose this hybridization since the classical ABC is limited in solving complex multimodal functions and functions with a narrow curving valley. QC would optimize the ABC since it reduces the number of individuals to represent the search space due to its quantum properties, such as superposition.

Each solution of the QABC is represented as a quantum vector. The algorithm starts by generating N random quantum vectors. The first step is to obtain a binary solution from the quantum representation by measuring the quantum vector, which is then evaluated using a fitness function. The next step is to perform a greedy selection and keep the solution with the best fitness value, and a quantum ABC operator is applied to obtain new solutions. The final step is to apply quantum interference, which allows shifting qubits towards the corresponding qubit representing the best-found solution.

The algorithm was tested against three benchmark functions, and its results were compared against other algorithms, such as quantum swarm optimization and a conventional evolutionary algorithm. The results demonstrated that it outperformed both algorithms and also was able to find the global optimum in two of the three functions tested.

Quantum Ant Colony Optimization Algorithm

Wang et al. [33] proposed a combination of the classical Ant Colony Optimization (ACO) algorithm with a Quantum-inspired Evolutionary algorithm (QEA), called Quantum Ant Colony Optimization (QACO), to solve a discrete binary optimization problem. The proposed algorithm introduces the qubit representation of a pheromone and the usage of quantum rotation gates.

The first step of QACO is to initialize the ant population, the number of generations and pheromone qubits with $\alpha = \beta = \frac{1}{\sqrt{2}}$, so that solutions have the same probabilities when generated. The second step is when each ant travels to the food source by observing the pheromone value. The third step is to calculate the fitness of each ant after building the solutions, and termination happens right after if the condition is met. The final step is to update the pheromone intensity using quantum rotation gate, and then a new iteration starts again.

Five different benchmark functions were used to test the algorithm's performance. The results were compared to the other two algorithms, the QEA and a Discrete Binary version of the particle

swarm optimization algorithm. The results showed that QACO effectively and successfully solved the benchmark functions and outperformed the other algorithms.

Quantum-inspired Swarm Evolution Algorithm

Huang et al. [19] proposed a Quantum-inspired Swarm Evolution Algorithm (QSEA). The goal of this proposal was to enhance an existing classical evolution algorithm (CEA). The algorithm was developed based on a Quantum-inspired Evolutionary Algorithm (QEA) and used swarm intelligence from a Particle Swarm Optimization technique. In this algorithm, a qubit is represented as $[\theta]$ and a quantum angle as θ.

Using the concept of swarm intelligence, qubits are regarded as an intelligent group called a quantum swarm. First, the best local quantum angle value and the best global value from local ones are found, so then according to these values, quantum angles are updated. The first step of the algorithm is to encode qubits using the quantum angle θ. The second step is modifying the QEA update procedure with improved particle swarm optimization formulae.

Experiments were conducted by solving three benchmark functions, and the performance was compared against a CEA. QSEA solved the functions effectively and outperformed the CEA in terms of quality and efficiency.

Quantum approaches using QUBO modeling

As mentioned previously in section 2.1.4, QUBO modeling is a popular approach for solving optimization problems using quantum computing. Martins et al. [23] developed a QUBO model to solve the Tail Assignment Problem (TAP). The TAP optimization is related to the ARP, with the main difference being that the TAP aims to produce a new flight schedule from scratch. In contrast, the ARP aims to quickly resolve a disruption by either delaying, canceling or swapping flights. The model proposed by Martins consisted of having each binary variable $q_{f,a}$ represent the assignment of flight f to aircraft a and the quadratic matrix contained values to penalize assignments that did not satisfy the proposed constraints. The problem was solved using D-wave's quantum computers with the quantum annealing algorithm, which produced successful results in a relatively short time.

Mohammadbagherpoor et al. [24] proposed a QUBO model to solve the airport gate scheduling problem. The problem is also related to aircraft assignments, but in this case, to airport gates. The QUBO was modeled by using binary variables $x_{i,k}$, which represents the assignment of parking aircraft i on gate k, to minimize the total walking distance from passengers arriving on the flights. The model was solved using Qiskit's quantum computers using the quantum approximate optimization algorithm (QAOA) with 27 qubits available, which required a reduced dataset to resolve the problem with the available memory.

2.3 Summary

This chapter presented background information and literature review regarding the airline's scheduling process, quantum computing (QC) and the mathematical model Quadratic Unconstrained Binary Optimization (QUBO). The airline scheduling process is an extensive process that begins days or weeks before the actual performance of the flights. During the execution of the flight plan, another process known as disruption management deals with resolving disruptions that may occur, including recovering aircraft, known as Aircraft Recovery Problem (ARP). QC is an emerging field that combines quantum mechanics with computers, aiming to obtain a more capable machine that can solve problems that are currently unfeasible for today's computers. Some companies offer access to their quantum computers via cloud programs, but there is a significant limitation regarding available qubits and time when using these computers. The QUBO model is widely used for optimization problems, consisting of mapping binary variables and assigning weights in a quadratic matrix. Different companies build quantum computers capable of solving problems modeled as a QUBO and different libraries for modeling this type of problem.

Chapter 3

Aircraft Recovery Problem

This chapter details the Aircraft Recovery Problem (ARP) and presents the dataset used for the problem. Section 3.1 describes the ARP and defines essential concepts that will be addressed through the remainder of the document. Finally, section 3.2 provides an analysis of the dataset used in this work.

3.1 Problem Description

The Aircraft Recovery Problem (ARP) was introduced in section 2.1.1 alongside the airline's scheduling process. The airlines' aircraft perform the flights scheduled in the previously generated flight plan. Still, due to different events, the flight plan might not be possible to be performed as scheduled and resulting in delays, cancellations or even aircraft swaps. As also previously mentioned in section 2.1.1, the ARP is part of the disruption management phase of the scheduling process. It consists of trying to recover the flight plan as fast as possible, with a low cost and minimizing the changes in the flight plan. The airline's operations control center (OCC) handles this problem and tries to quickly obtain a feasible and low-cost solution by swapping aircraft, delaying or even canceling flights, followed by recovering the crew and the passengers.

One of the main problems regarding the ARP is that a single affected resource may lead to a chain of events that will affect many subsequent flights. This problem can scale to thousands of passengers depending on the impact of the disruption [36] which translates to an even more significant financial loss for the company. In the past few years, airlines had a lower volume of flights due to the COVID-19 pandemic, which affected global tourism and business travels and resulted in a substantial financial loss for airlines, when most were grounded [12]. These events caused even more pressure for companies to avoid financial losses, making the ARP an even more critical step to reducing costs [15].

The remainder of this section defines essential concepts regarding the ARP used during the remainder of the document. It also defines the main constraints considered for supplying a solution,

presents the costs associated with each flight and introduces the optimization criteria for obtaining a low-cost solution.

3.1.1 Operational Plan

Airline scheduling is the leading planning activity involving all the company's valuable resources and generating an operational plan. It includes the flights performed, with associated departure and arrival information, the used aircraft, and the crew operating [18]. The scheduling is performed by solving an optimization problem given the company's resources, requirements, and constraints to find feasible flight assignments that maximize the airline's profit [13] and deliver an operational plan. Nowadays, most airlines follow the concept of having a central hub, which is the location where the companies' resources are primarily available, where the maintenance is performed and where most of their flights arrive and depart from [6].

3.1.2 Events and Disruptions

In the context of this dissertation, events are unscheduled activities that affect the usage of one or more aircraft during a specific period. Unscheduled events, known as disruptions, are unpredictable occurrences that may or may not impact the airline's scheduled flight plan. Airports Council International (ACI) defines flight disruptions as [1]: *"situations where a scheduled flight is canceled, or delayed for two hours or more, within 48 hours of the originally scheduled departure time"*. These events can have several causes, the most common being bad weather, unscheduled maintenance or even airport closures and may require a rescheduling of the current flight plan.

Events can be classified as *hard-change* or not, depending on the disruption's cause. The *hard-change* classification means that there is no possible alternative to avoid delaying or canceling a group of flights, which an airport closure, for example, can cause. This type of event usually requires a more complex solution since it affects more than one aircraft, and a cost regarding the total delay of each affected flight is unavoidable. When the event is not considered *hard-change*, the flight delay can be prevented by swapping aircraft if the total cost is lower than delaying the flight.

3.1.3 Constraints

When creating a model that can solve the ARP, it is crucial to define constraints to ensure that the generated solutions are feasible and valid for real-scenario usage. These constraints relate to the flight plan, aircraft and airport, and can be summarized in the following points:

- Each flight should be assigned to precisely one aircraft unless the flight is canceled
- The aircraft performing the flight must be at the flight's departure airport, even if it requires delaying

- The aircraft performing the flight should also be able to transport the flight's original number of passengers and have enough range to travel the required distance
- For an aircraft to perform two consecutive flights, it must satisfy a minimum turn time between them

3.1.4 Costs

The performance of a flight involves many different costs. These costs are applied to all flights but vary in value, especially regarding the aircraft performing the flight. Bigger aircraft usually require a more considerable amount of fuel, parking space and handling when compared to smaller aircraft. The costs associated with performing a flight considered for this work include: the Air Traffic Controller (ATC) cost, fuel, landing, parking and handling. Equation 3.1 illustrates the calculation of a flight's total cost with the different costs associated regarding the flight f and the aircraft a performing it. Figure 3.1 illustrates the different costs of performing a flight through its different stages also including costs regarding the ARP such as the aircraft swap and delays cost. All these values are available in the dataset used, which will be introduced in section 3.2.

$$cost_{f,a} = ATC_f + fuel_{f,a} + landing_a + parking_a + handling_a \tag{3.1}$$

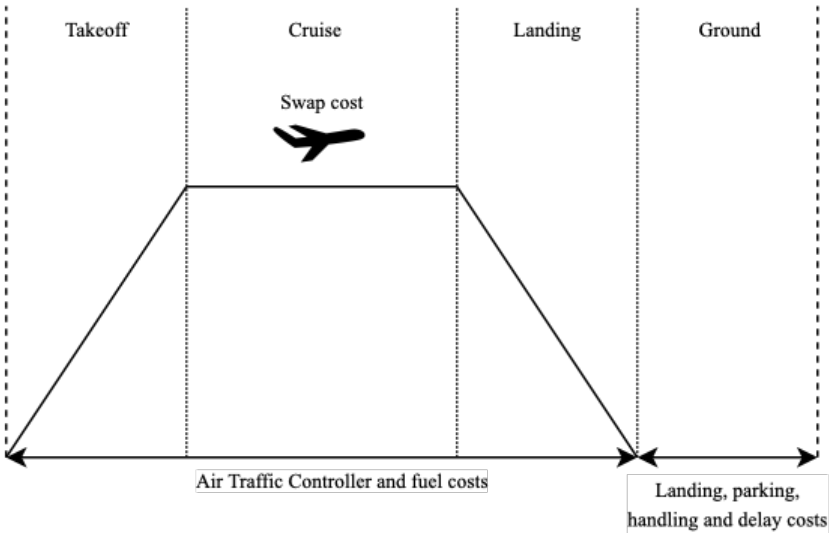


Figure 3.1: Cost distribution through different flight stages

3.1.5 Optimization Criteria

As mentioned in the beginning of this section, the main goal of the ARP is to recover the flight plan while minimizing the total changes to the flight plan while minimizing the cost of the process. Other criteria should also be considered, such as efficiently using every available aircraft since they do not generate revenue when not in use. In the resolution of the ARP, the main criteria that will be followed are then:

- Minimize the number of aircraft swaps
- Minimize the total delay
- Minimize the total cost
- Minimize the execution time to obtain a solution
- Maximize the usage of each aircraft

3.2 Dataset Analysis

The dataset used in this work contains real data from TAP Air Portugal[1], which is Portugal's leading airline with its main hub in Lisbon. The dataset contains information regarding the airline's operational plan during September 2009 and is composed of 6 tables: *Aircraft*, *Aircraft Models*, *Airport Charges*, *City Pairs*, *Events* and *Flights*. A brief summary of the content of the dataset can be found in table 3.1.

Table	**Lines**	**Columns**
Aircraft	72	3
Aircraft Models	9	14
Airport Charges	312	4
City Pairs	210	10
Events	49	26
Flights	8069	14
Total	**8721**	**71**

Table 3.1: Original dataset summary

The original dataset contains much more information not relevant to the ARP, so it was decided to remove unused columns and aggregate related tables to facilitate the coding process. The final resulting dataset is composed of three tables: *Aircraft*, *Flights* and *Events*. The *Aircraft* table contains information regarding each aircraft available in TAP's fleet, the *Flights* table contains data regarding each flight performed during September 2009 and finally, the *Events* table provides the different disruptions that happened during the same period.

A brief summary of the contents of the reduced dataset is shown in table 3.2, and the complete composition and description of the three tables introduced previously are presented in the following sections. The data analysis will be performed on the aggregate dataset since it contains the relevant information for the problem.

3.2.1 Aircraft Analysis

Table 3.3 presents the *Aircraft* table composition, including the different columns, types, and a description of each value.

[1] https://www.flytap.com/

Table	Lines	Columns
Aircraft	72	9
Events	49	7
Flights	8069	14
Total	**8190**	**30**

Table 3.2: Aggregate and reduced dataset summary

In total, there are nine different aircraft models. These aircraft models are divided into narrow-body (NB) and wide-body (WB) categories, that contains three and six models, respectively. Table 3.4 illustrates the different aircraft models present in the dataset and their respective fleet type.

According to the International Civil Aviation Organization (ICAO), NB aircraft contain only one aisle dividing the seats into two axial groups. At the same time, WB is more extensive, having two aisles dividing the cabin into three axial groups [20]. NB aircraft are generally used for shorter flights due to their maximum flight range, while WB aircraft are used for flights with longer distances since they have a more extensive flight range.

The total fleet comprises 72 aircraft, of which 17 of them are WB, and the remaining 55 are NB, with an approximate distribution of 25% and 75%, respectively, as shown in figure 3.2.

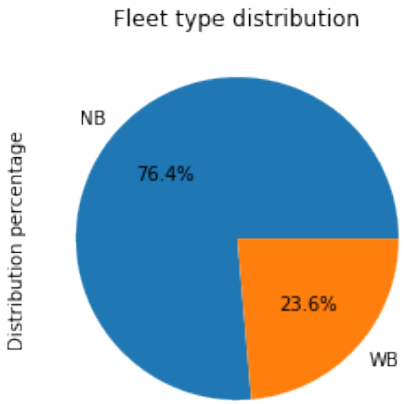


Figure 3.2: Aircraft fleet distribution percentage

As previously mentioned, the fleet is composed of nine different aircraft models. These models have unique attributes and capacities. Figure 3.3 illustrates the different passenger capacity that each model has. On average, around 150 passengers can be transported each flight, but the difference between capacities is still considerable. An Airbus A340 may transport over 250 passengers, while a Beechcraft can only carry less than 25 passengers.

3.2.2 Flights Analysis

Table 3.5 presents the *Flights* table composition, including the different columns, types, and a description of each value.

Column	Type	Description
tail_number	string	Unique identifier of an aircraft
model	string	The model of the aircraft
fleet	string	Defines if the aircraft is wide or narrow bodied
capacity	int64	Passenger capacity of the aircraft
atc_avg_cost_nautical_mile	float64	Average cost of the Air Traffic Controller per nautical mile
maintenance_avg_cost_minute	float64	Average cost per minute of the aircraft's maintenance
fuel_avg_cost_minute	float64	Average cost of the fuel consumed per minute of flight
airport_handling_cost	int64	Ground cost for handling the aircraft on the airport
max_range_in_km	float64	Aircraft's maximum flight range in kilometers

Table 3.3: Aircraft table description

Model	Fleet
A310	WB
A319	NB
A320	NB
A321	NB
A330	WB
A340	WB
BEH	NB
ER4	NB
F100	NB

Table 3.4: Aircraft models and fleet types

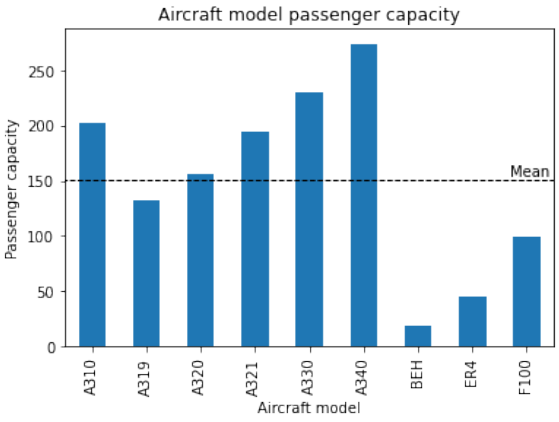


Figure 3.3: Aircraft model passenger capacity

Column	Type	Description
flight_date	datetime	Flight's departure date
flight_number	int64	Unique number identifying the flight
origin	string	Origin airport
destination	string	Destination airport
scheduled_time_of_departure	datetime	Expected time to depart
scheduled_time_of_arrival	datetime	Expected time to arrive
tail_number	string	Scheduled aircraft's tail number to perform flight
total_seats_sold	int64	Number of passengers in flight
distance_in_km	float64	Flight's distance in kilometers
distance_in_nautical_miles	float64	Flight's distance in nautical miles
NB_LND_amount	float64	Cost for landing a narrow body aircraft in the destination
NB_PRK_amount	float64	Cost for parking a narrow body aircraft in the destination
WB_LND_amount	float64	Cost for landing a wide body aircraft in the destination
WB_PRK_amount	float64	Cost for parking a wide body aircraft in the destination

Table 3.5: Flights table description

There are over 8000 flights that compose the month's flight schedule. As it can be observed in figure 3.4, flights are distributed throughout the month, having an average of around 260 flights per day. Still, as observed in the figure, there are days in which over 300 flights are operated and other days only around 100 are operated. This analysis shows that depending on the day a disruption happens, there might be more or less complexity in obtaining a solution to the problem.

As mentioned in section 3.2.1, aircraft are split into NB and WB categories, which are usually assigned to shorter and longer flights, respectively. Figure 3.5 illustrates the fleet distribution over all flights and also the average flight distance based on the aircraft's fleet. It is possible to conclude that around 90% of the flights are relatively short and do not require WB aircraft. When rescheduling the flight plan, NB aircraft cannot perform more extended flights that are scheduled initially to a WB aircraft due to their limited flight range. This restriction may reduce the complexity of the solution since this is not a feasible option.

In total, the dataset contains 76 different departure and arrival airports. Still, every airline has its central hub, where most of its flights are operated. In this case, Lisbon is the main airport used by TAP, with over 3000 flights being operated there, composing at around 40% of the monthly flight plan. Figure 3.2 illustrates the five main departure airports from the dataset. Suppose a disruption happened in the airline's central hub. It should be easier to obtain a solution since most of its operational plan is present in the airport, and more options to swap if needed are available.

3.2.3 Events Analysis

Table 3.6 presents the *Events* table composition, including the different columns, types, and a description of each value.

As mentioned previously, events are unexpected occurrences that may disrupt the original flight plan and require changes to recover the operational plan. The dataset comprises 49 events

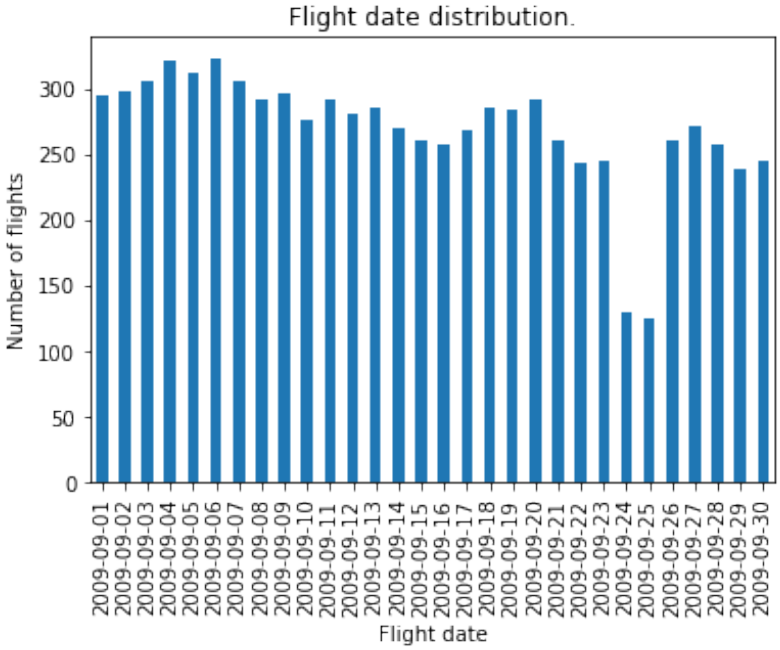
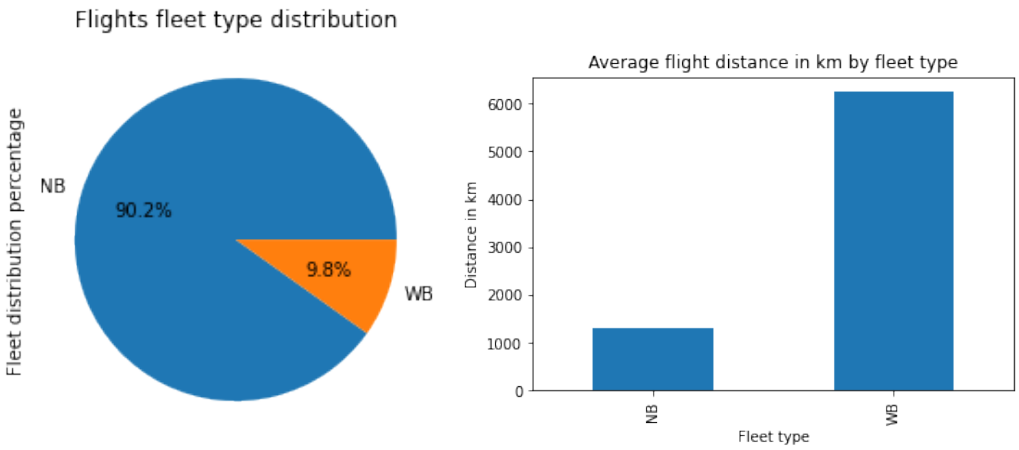


Figure 3.4: Flights date distribution



(a) Flights fleet distribution (b) Average flight distance based on aircraft fleet

Figure 3.5: Flights distribution

Column	**Type**	**Description**
event_id	int64	Unique id for the disruption
event_time	datetime	Time when disruption begun
resource_affected	string	The first affected aircraft by the disruption
flight_date	datetime	The first affected flight by the disruption
flight_number	int64	The flight number of the first affected flight
end_time	datetime	Expected time for the disruption to be resolved
hard_change	bool	Determines if the disruption affects one or more resources

Table 3.6: Events table description

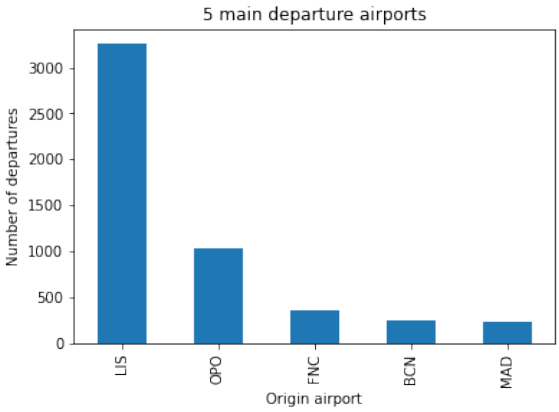


Figure 3.6: Top 5 main departure airports

with a total of 13 different causes. These causes may affect the aircraft, airport or the flight crew, and are listed as follows:

AIRP: This cause relates to an airport closure.

ATC: This cause relates to problem regarding the Air Traffic Controller (ATC) communications.

COMM: This cause relates to communication issues regarding the aircraft.

CROT: This cause relates to a last minute crew rotation required for the flight.

HAND: This cause relates to an aircraft's handling issues.

INDUTY: This cause relates to problems regarding the crew in duty of the flight.

MAINT: This cause relates to unexpected maintenance that needs to be performed on the aircraft.

METEO: This cause relates to bad weather which disrupts the airspace.

OTHER: This relates to other causes that needs the aircraft to be grounded.

ROT: This cause relates to aircraft rotation issues.

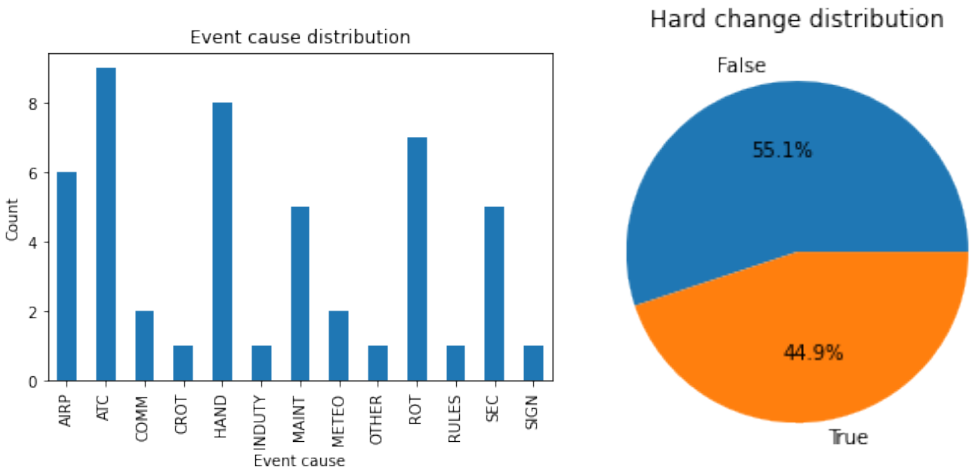
RULES: This cause relates to rules applied to an aircraft that requires it to be grounded.

SEC: This cause relates to security issues that may relate to the aircraft or the airport.

SIGN: This cause relates to signal malfunctioning in the aircraft.

Of all these causes, four of them ('AIRP', 'ATC', 'METEO' and 'SEC') provoke a disruption that may affect more than one aircraft and require the affected resources to be grounded until the event is over. The event causes and hard changes distribution can be visualized in figure 3.7. By

analyzing the figures, it is possible to conclude that slightly over half of the disruptions affect one resource, which tends to have a less complex and costly solution since most of its aircraft are not affected.



(a) Event cause distribution (b) Hard change distribution

Figure 3.7: Event causes and hard changes distribution

Figure 3.8 illustrates the date distribution of the events that happened during the monthly flight plan. It is possible to conclude that disruptions can occur almost daily, with the possibility of having five disruptions in one day. This observation shows why the ARP is a complex problem that requires low-cost and fast solutions since it may be needed to reformulate the flight plan daily.

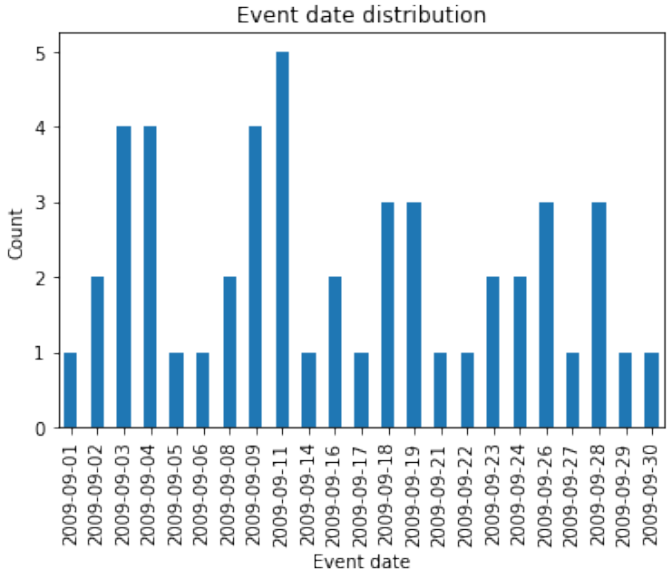


Figure 3.8: Event date distribution

3.3 Summary

In this chapter, the scope of this dissertation was provided by further introducing the Aircraft Recovery Problem (ARP). Critical concepts regarding the ARP, which are used throughout this document, such as constraints and costs, were also introduced. Lastly, the dataset used in this work was introduced and analyzed by presenting essential details regarding the aircraft, flights and events that it contains.

Chapter 4

Quantum Approach to Flight Rescheduling

This chapter details the approach to solving the Aircraft Recovery Problem with quantum computing. Section 4.1 presents the approach adopted, which consists of a two-stage algorithm. Section 4.2 details the aircraft feasible flights graph which was used for generating possible flight assignments used in the model. Section 4.3 introduces the modeling of this problem in the form of a Quadratic Unconstrained Binary Optimization (QUBO).

4.1 Two Stage Algorithm

The proposed solution will be a Two-Stage Heuristic Algorithm based on Zhang's [35], presented in section 2.2.1. Initially, all feasible Lines of Flights (LOFs) for each aircraft in the dataset will be generated. The first stage of the algorithm will score each of these LOFs and remove the ones with an evaluation below a defined threshold to minimize the problem's solution space. The new reduced set of LOFs will be used as an input for the second stage of the algorithm.

The main difference between this algorithm and the one proposed by Zhang will be in the second stage. During the second stage, a quantum algorithm will be applied to solve the problem and provide a solution, which is expected to provide a better solution and provide it faster. It is important to note that a new quantum algorithm will not be developed in this dissertation, but rather one of the algorithms presented in section 2.2.2 will be used. Figure 4.1 presents a visual representation of the proposed algorithm.

4.1.1 Initialization

The algorithm initializes by receiving a disruption and localizing the first affected flight in the original schedule. The next step is to check the disruption's cause and determine if a hard change is required, so then it can apply the delay to each flight affected until the disruption is over.

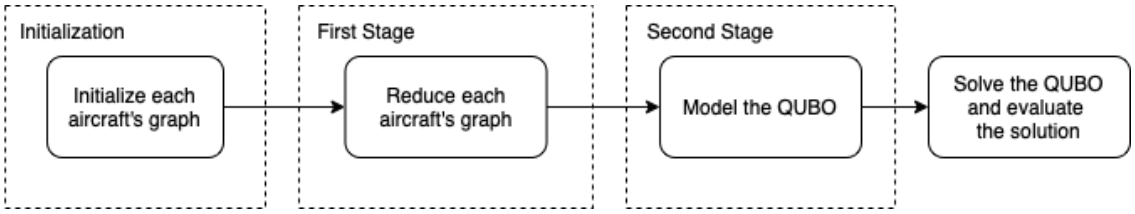


Figure 4.1: Proposed algorithm overview

The initialization process proceeds to generate a graph for each aircraft and, starting at the first disrupted resource, of all the possible paths they can perform. However, to do so, it is necessary to obtain the previous flight each aircraft was performing before the disruption to obtain the same airport in which the graph starts. Section 4.2 presents the graph structure and generation process with more details.

After all the steps presented in this subsection are completed, the algorithm proceeds to initialize the *First Stage*, which is detailed in the following subsection. An overview of this initial stage is presented in figure 4.2.

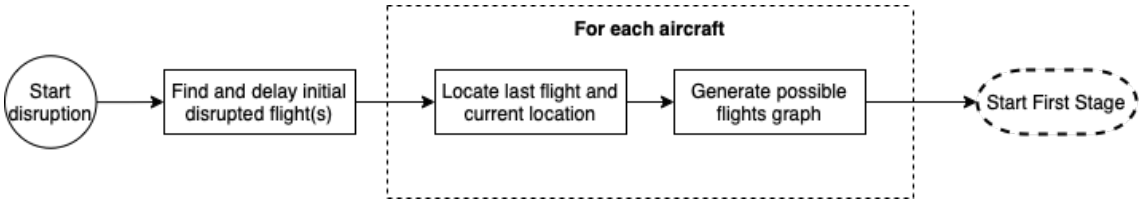


Figure 4.2: Initialization stage overview

4.1.2 First Stage

The goal of the algorithm's first stage is to reduce the complexity of the problem due to its large amount of possible combinations between flights and aircraft. To do so, it is necessary to iterate through each aircraft's feasible flights graph and remove paths that are the most costly. By removing costly paths the complexity of the problem is reduced, since each binary variable of the program represents the assignment of a flight to an aircraft and are added to the program by iterating through each aircraft's graph.

When iterating through each aircraft's graph, each edge's delay is analyzed and compared to a defined value θ, in this work considered as 240 minutes, which when exceeded the edge is removed and consequently the next node as well, if no other edge is connected to it.

4.1.3 Second Stage

The second stage of the algorithm consists of modeling the problem as a QUBO and solving it using quantum and classical solvers. Section 4.3 provides a more detailed explanation of the modeling of the problem.

The decision to solve the problem by modeling it into a QUBO was made because different quantum libraries, such as IBM's Qiskit and D-wave's Leap, already provide tools to model QUBO problems and provide different solvers to solve them. These different options meant that it was only necessary to have one model and be able to use different solvers from different libraries for the same problem. Another reason to model the problem as a QUBO was that many previous optimization problems solved with quantum computing were modeled the same way.

After modeling the program into a QUBO, different solvers were used to solving the problem. Since many quantum solvers have a limited number of qubits or time usage limit, a classical solver was needed to speedup the coding process while also validating the results. IBM's CPLEX Optimizer[1], a classical solver that has built-in integration with Qiskit, was used as the solution to this problem.

Different quantum solvers were used to solve the problem. Qiskit provides different algorithms to be used in their quantum solvers, such as Quantum Approximate Optimization Algorithm (QAOA) and Variational Quantum Eigensolver (VQE) algorithm. D-wave offers quantum and hybrid solvers, which are capable of resolving quadratic programs using Quantum Annealing. A detailed comparison and analysis of the results provided by each solver is detailed explained in chapter 5.

4.2 Aircraft Feasible Flights Graph

A graph containing all the feasible flights for each aircraft is built when a disruption occurs. The main goal of the graph is to provide alternative flight sequences for the entire fleet and find possible changes in the flight plan to resolve the disruption at the lowest possible cost. The graph is also used to model the problem into a QUBO since it contains every possible assignment of flights to aircraft and the delays for performing flight sequences.

4.2.1 Graph Structure

The aircraft graphs were built using *networkx*'s[2] *MultiDiGraph* module, which allows the creation of a directed graph with multiple edges between two nodes. Each node in the graph represents a flight that the aircraft can perform, and each edge represents a connection between two flights that the aircraft can perform sequentially. An example of a graph generated for an Airbus A319 is shown in figure 4.3.

Each edge contains the delay in performing the two sequential flights, which considers the actual arrival time and the scheduled arrival time from the previous flight and the minimum rotation time. Equation 4.1 presents the formula used to calculate the delay for performing flight f' and f sequentially.

[1] https://www.ibm.com/analytics/cplex-optimizer
[2] https://networkx.org

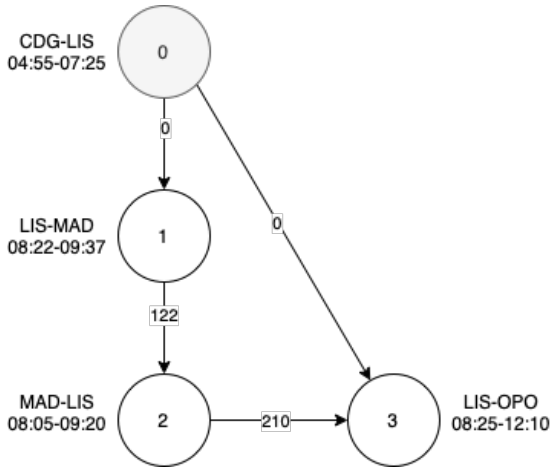


Figure 4.3: Generated graph for an Airbus A319

$$\begin{aligned} AAT_{f'} &= SAT_{f'} + TD_{f'',f'} \\ PD_{f',f} &= max(AAT_{f'} + MTT - SDT_f, 0) \\ TD_{f',f} &= PD_{f',f} + DD_f \end{aligned} \tag{4.1}$$

The variable $AAT_{f'}$ represents the actual arrival time of the last flight, calculated by adding the scheduled arrival time ($SAT_{f'}$) with the total delay from the flight before it ($TD_{f'',f'}$). $PD_{f',f}$ refers to the propagated delay when performing flights *f'* and f sequentially. It is calculated by taking the previous flight's actual arrival time, adding the minimum turn time (MTT), and subtracting the next flight's scheduled departure time SDT_f. If the resulting value is smaller than or equal to zero, there is no propagated delay for performing both flights. The variable $TD_{f',f}$ represents the total delay for performing both flights, and it is calculated by the sum of the propagated delay and the disruption delay for the following flight (DD_f). The resulting value of $TD_{f',f}$ is then added to the edge between nodes *f'* and *f*.

Since there are many possible paths that an aircraft can perform, it might be needed to add two or more edges between two nodes. Figure 4.4 illustrates why multiple edges are needed, since there are two possible paths between the starting node and node 4, with different delays: {0, 3, 4} (A) and {0, 1, 2, 3, 4} (B). Both paths are valid, but path B required a 5-minute delay to perform flight 4 after flight 3. The module *MultiDiGraph* used to build the graph requires a unique key to represent each edge between two nodes, so it was decided to use the path starting at node 0 until the current node, which can also be visualized in figure 4.4.

4.2.2 Generation Process

The first step when generating the graph is to find each aircraft's latest flight that departed before the disruption happened and use it as the graph's root node. Obtaining the last flight is an essential step so that it is possible to determine each aircraft's current location to start building the feasible flights graph. After concluding the first step, the graph is built recursively using the list of flights

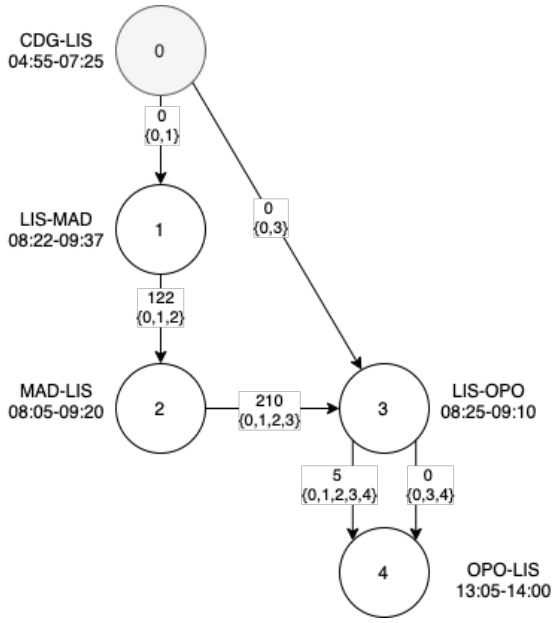


Figure 4.4: Generated graph for an Airbus A319, including edge keys and the case of more than one edge between two nodes

starting from the first disrupted one. Listing 4.1 presents a simplified version of the code used to generate the graph. Line 5 contains the function *calc_delay*, which calculates the delay for performing two consecutive flights as presented in equation 4.1.

The function *build_graph* receives four arguments which are the last or initial flight, the list of flights starting with the first disrupted one, the path, which initially is an empty list and the propagated delay, which is initially set to 0. The first statement of the function is to add the last flight to the current path so that it will be used as a key to the added edges. Then, the entire list of flights is iterated to find all consecutive flights to the last one. When a flight is found, the function calculates the delay in performing both and creates an edge between them. Finally, the function is called again, but now using the newfound flight as the first argument, the list of flights is reduced to start after the new flight and the edge's delay is now passed as the propagated delay.

```
1 def build_graph(last_flight, flights, path, propagated_delay):
2     path.append(last_flight)
3     for f in flights:
4         if are_consecutive(last_flight, f):
5             delay = calc_delay(last_flight, f, propagated_delay)
6             add_edge(last_flight, f, key=path, delay=delay)
7             build_graph(f, flights[1:], delay)
```

Listing 4.1: Partial code for generating an aircraft's graph

4.3 Quadratic Unconstrained Binary Optimization Modeling

The problem is modeled into a QUBO during the second stage of the algorithm. Therefore it is necessary to define what will the binary variables represent. Based on other previous quantum solutions of similar optimizations problems, such as the Tail Assignment Problem by Martins et al. [23], it was decided that the binary variables would follow the format $q_{f,a} \in \{0,1\}$ where $q_{f,a} = 1$ represents the assignment of flight f to aircraft a. The QUBO model is set as a sum of different penalties and the objective function, which is represented in equation 4.2. P_a represents the assignment penalty, P_b the impossible pairing penalty, P_{di} the initial delay, P_s the swap penalty, P_{dq} the quadratic delay and finally, C_{OF} represents the cost of the objective function.

$$C = P_a + P_b + P_{di} + P_s + P_{dq} + C_{OF} \tag{4.2}$$

Qiskit's optimization module[3] was used to model the problem with its *QuadraticProgram* class, which allows the creation of a QUBO model that can be solved using Qiskit's solvers or by converting and using D-wave's solvers. To add the binary variables to the problem, each aircraft's graph is iterated to obtain the flights that it can perform and add one binary variable at time following the convention $q_{f,a}$. For the remainder of this section, the following notations were considered when building the model used to solve the ARP:

A : set of all aircraft

S : set of all flights composing the original schedule

$F \subset S$: set of all flights, starting in the first disrupted flight, to be used in the model

$A_f \subset A$: subset of A that can perform a given flight f

$F_a \subset F$: subset of F that can be performed by a given aircraft a

$F_s \subset F_a$: subset of F_a where the aircraft a is not originally assigned to perform the flights

$F_i \subset F$: subset of F that represents the initially affected flights when the disruption occurred

$I_f \subset Fa$: subset of Fa that cannot be assigned simultaneously with flight f

$DC_f \subset Fa$: subset of Fa that is directly connected to a given f on the corresponding aircraft's connection network graph

$IC_f \subset Fa$: subset of Fa that is indirectly connected to a given f on the corresponding aircraft's connection network graph

δ : corresponds to the delay penalty factor

σ : corresponds to the swap penalty factor

[3] https://qiskit.org/documentation/optimization/apidocs/qiskit_optimization.html

4.3.1 Penalties

As explained previously in section 2.1.4.2, a QUBO is unconstrained. Still, there is the need to penalize some cases to obtain a feasible and low-cost solution. The penalty factors are translated from a classical constraint as presented in table 2.1, and their values are added into the final objective function.

4.3.1.1 Assignment Constraint

When obtaining a flight plan, precisely one aircraft must perform any given flight. This constraint is ensured by applying a quadratic penalty to schedules that do not comply, as shown in equation 4.3.

$$P_a = \sum_f^F \left(\sum_a^{A_f} q_{f,a} - 1 \right)^2 \tag{4.3}$$

This penalty adds a positive value for occurrences where a flight f is not performed by exactly one aircraft, increasing the final cost of the solution.

4.3.1.2 Impossible Pairing

This constraint penalizes the simultaneous assignment of flights f and f' to an aircraft a when there is no valid path between both flights. Equation 4.4 defines the quadratic penalty function for this constraint.

$$P_b = \sum_a^A \sum_f^F \sum_{f'}^{I_f} q_{f,a} q_{f',a} \tag{4.4}$$

This penalty function adds the value of 1 when both flights f and f' are performed by the same aircraft a, which increases the total cost of the solution.

4.3.1.3 Initial Delay Penalty

The disruption initially affects one or more flights, which must be delayed until the end of the disruption if they will be performed. The delay formula is show in equation 4.5, where δ is the delay for flight f which was caused due to the disruption.

$$P_{di} = \sum_f^{F_i} \sum_a^A q_{f,a} \delta \tag{4.5}$$

4.3.1.4 Swap Penalty

The original schedule already contains one aircraft assigned to each flight, and in the cases that there is a need to swap the original aircraft, a penalty must be applied. Equation 4.6 illustrates the penalty applied when a swap occurs, with σ being the penalty factor.

$$P_s = \sum_{a}^{A} \sum_{f}^{F_s} q_{f,a} \sigma \tag{4.6}$$

4.3.1.5 Quadratic Delay Penalty

When the original graph is created for each aircraft, many different flight combinations are available for them to perform. Still, if an aircraft performs two consecutive flights, there might be a delay when performing both of them. For this, it is necessary to add a quadratic penalty to penalize cases when this delay may occur, as shown in equation 4.7. The function *delay(f,f',a)* returns the delay, in minutes, for aircraft a to perform both flights f and f' which is calculated as presented previously in equation 4.1. The value σ represents the delay penalty factor that multiplies the schedule's total delay in minutes.

$$P_{dq} = \sum_{a}^{A} \sum_{f}^{F_a} \sum_{f'}^{DC_f} q_{f,a} q_{f',a} delay(f, f', a) \delta \tag{4.7}$$

Special Case

As presented by the end of section 4.2.1, two nodes might have more than one edge between them due to different delays from different paths. When adding the quadratic delays, in the case of one or more edges between the two flights, the function presented in 4.7 is not used since there may be different values for the delay when performing flights f and f'.

Each edge's unique key solves the conflict of more than one edge between two nodes since it contains the entire path. The goal is to iterate each path backwards simultaneously and find the first flight f'' that is not present in each edge's key, so then the delay calculation function is set as $delay(f'', f', a)$ in equation 4.7, using flights f'' and f'. By observing figure 4.4, the special case occurs with flights 3 and 4. Instead of using the flight pair's (3, 4) delay, it should use the pairs (2,4) with the value 5 and (0,4) with the value 0.

4.3.2 Objective Function

Since there is a considerable difference in the total operational cost of the flights, including the aircraft performing it, it was decided to normalize the values in the range [0, 1]. With the normalization, each flight has the same weight on the final cost of the operational schedule. Equation 4.8 illustrates the process of the normalization, which used the Min-max normalization method. $cost_{f,a}$ represents the cost for flight f to be performed by aircraft a, which was introduced in section 3.1.4, $mincost_f$ represents the minimum performance cost for flight f to be performed by any aircraft and $maxcost_f$ represents the maximum performance cost for flight f to be performed by any aircraft.

$$C_{OF} = \sum_{f}^{F} \sum_{a}^{A_f} \frac{cost_{f,a} - mincost_f}{maxcost_f - mincost_f} q_{f,a} \tag{4.8}$$

4.3.3 Quadratic Matrix

The penalties and the objective function presented in this section are used for generating the quadratic matrix. To facilitate the visualization, figure 4.5 illustrates a upper triangular quadratic matrix partially generated for solving a disruption. The figure contains colored and uncolored cells. The uncolored cells are composed of the value 0. The colored cells represent the resulting values from equations 4.3, 4.4 and 4.7, matching the respective equation's color. The other previously presented equations also contribute to obtaining the values in the matrix, but three different equations were used in the figure to facilitate its visualization.

	0,0	0,1	0,2	1,0	1,1	1,2	2,0	2,1	2,2
0,0	-0.94	2	2	1			0.01		
0,1		-0.42	2		1			1	
0,2			-0.5			0.12			1
1,0				-0.44	2	2	1		
1,1					-1	2		1	
1,2						-0.5			0.2
2,0							-0.33	2	2
2,1								-0.48	2
2,2									-1

Figure 4.5: Partial representation of a quadratic matrix generated using the penalties and objective function. The rows and columns keys are in the format *f,a*, where *f* represents the flight and *a* represents the aircraft performing it.

4.4 Summary

This chapter presented the approach used to model the Aircraft Recovery Problem (ARP) into a problem that can be solved using quantum computing (QC). A Two-Stage Algorithm was introduced, along with the aircraft's feasible flight graph used in the modeling process. The chapter also introduced the modeling of the problem into a Quadratic Unconstrained Binary Optimization (QUBO), which was chosen due to the available libraries that can model optimization problems as such and existing literature regarding this model. The objective function and the different penalties were presented and represented in a matrix form in figure 2.1.

Chapter 5

Experiments and Results

This chapter provides an overview of the experiments conducted when solving the ARP using classical, hybrid and quantum solvers and an analysis of the results obtained. Section 5.1 details all the experiments conducted. Section 5.2 provides the metrics used to analyze and compare different solutions in each experiment. Finally, section 5.3 analyzes the results obtained with different solvers while comparing them to other previous implementations.

5.1 Experiments Description

Two main experiments were conducted in this study. The first one was to compare the results and time of this solution against a previous implementation by Ferreira [16], presented in section 2.2.1, which used the same dataset as in this work. The second one was to compare the different available solvers and algorithms capable of resolving QUBO models, some being classical and others quantum. It is also important to mention that all experiments were conducted on the same Apple MacBook Pro equipped with an M1 Pro ARM CPU chip with 10-Cores and 16GB of RAM.

5.1.1 First Experiment Description

As previously mentioned, Ferreira [16] developed two algorithms, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), to solve the ARP and used the same dataset available in this work. Using the same dataset allows for a valid comparison between each solution. Five different disruptions were solved in Ferreira's experiments, and these disruptions relate to flights 928, 1917, 864, 1614 and 839, which are not classified as *hard-change*. Table 5.1 illustrates the primary details regarding each disruption, such as the airport where it occurred and the aircraft model affected.

This first experiment's goal is to compare Ferreira's and this work's implementations in five different scenarios, which correspond to each of the disruptions presented in table 5.1 since they were solved in Ferreira's work and the results are available for comparison. It was chosen to use

Flight	Airport	Aircraft
928	LIS (Lisbon, Portugal)	A319 (NB)
1917	LIS (Lisbon, Portugal)	A319 (NB)
864	LIS (Lisbon, Portugal)	A320 (NB)
1614	FNC (Madeira, Portugal)	A319 (NB)
839	FCO (Roma, Italy)	A320 (NB)

Table 5.1: First experiment disruptions details

Ferreira's PSO algorithm in this experiment due to its better performance in terms of time and cost when compared to the ACO. The PSO algorithm will be tested against this implementation using D-wave's *LeapHybridSampler*, a hybrid solver, to assess the effectiveness of quantum computing in solving the same problem. The metrics chosen for comparison in this experiment are presented in section 5.2.1 and the results are found in section 5.3.1.

5.1.2 Second Experiment Description

The ARP was modeled as a QUBO problem in this work, and as mentioned in section 2.1.4, there are different classical and quantum solvers capable of solving this type of problem. This experiment aims to compare the performance and solutions of classical and quantum solvers to resolve the ARP modeled as a QUBO. Five different disruptions were chosen for this experiment and are related to flights 1, 2 and 3. Some of these disruptions are considered *hard-change* and require a more complex solution. The solution comparisons will be related to the total delay, swaps, cancellations and execution time to compare their performances. The classical solver used is IBM's CPLEX, and the quantum solver is D-wave's *LeapHybridSampler*. It is important to mention that *LeapHybridSampler* is a hybrid solver, meaning it solves problems using both quantum and classical computers, since an actual quantum computer with enough qubits is not available for usage.

Flight	Airport	Aircraft
712	LIS (Portugal)	A319 (NB)
686	LIS (Portugal)	A319 (NB)
152	REC (Brazil)	A330 (WB)
139	LIS (Portugal)	A330 (WB)
916	OPO (Portugal)	A319 (NB)

Table 5.2: Second experiment disruptions details

The chosen disruptions were regarding flights 712, 686, 152, 139 and 916 due to their different attributes and are presented in table 5.2. Flights 712, 686 and 152 are classified as hard-change, meaning that delaying the affected flights is unavoidable, as detailed in section 3.1.2. Another essential attribute regarding flights 152 and 139 is that both aircraft performing are part of the wide-body (WB) fleet. The airline's fleet is composed mainly of narrow-body (NB) aircraft, which

usually cannot perform long-haul flights assigned to WB aircraft, leaving fewer options for swapping the affected WB aircraft. The main objective of this experiment is to analyze how each solver performs when resolving the ARP regarding different types of disruptions and affected aircraft.

5.2 Metrics

This section presents the metrics available and the selected ones for comparison in both experiments introduced in the previous section. The unit of measurement of each metric will also be presented in this section.

5.2.1 First Experiment Metrics

In Ferreira's [16] work, five different disruptions were solved as presented in section 5.1.1. The data available in his work for resolving these disruptions are each algorithm's total execution time, total operational cost, total delay and utility of the solution. It was chosen to use only the total execution time and delay as a metric when comparing Ferreira's implementation against this work's. Regarding measurement units, the solution's total execution time is expressed in milliseconds and the delay is expressed in minutes.

The solution utility was not implemented in this work, so there is no possibility to compare it. Regarding solution cost, this work modeled the problem as a Quadratic Unconstrained Binary Optimization (QUBO) and used normalization for the objective function, which modifies the range of each flight's cost to a value between 0 and 1, differing from Ferreira's, which does not allow for a valid comparison.

5.2.2 Second Experiment Metrics

As presented in section 5.1.2, the experiment also compares the performance regarding five different disruptions. Since the same QUBO model was used in this experiment but with different solvers, more metrics can be considered. This work's solutions provide the total execution time, number of swaps, number of cancellations and total delay, and will all be used as metrics for comparing the different solvers. The solution's total cost, which is based on the other metrics, is also available. In terms of units of measurement, the total execution time is represented in milliseconds, the total delay in minutes, and swaps and cancellations are represented in units.

5.3 Result Analysis

This section provides an analysis of the results regarding the two previously presented experiments. Some solvers used are cloud-based and take considerably more time in the program due to internet delays and queues in the cloud to solve the problem. Each of these cloud solvers provides the internal execution time in its solution, which was used as the final time in the analysis below to provide better comparisons against other solvers. D-wave provides the Quantum Processing Unit

(QPU) usage time in the solution[1], which contains the exact execution time of the program without including network delays and execution queues waiting time. Figure 5.1, obtained from D-wave's operation and timing documentation, illustrates the process of timing the solver and includes all the steps the problem goes through until it is returned to the client.

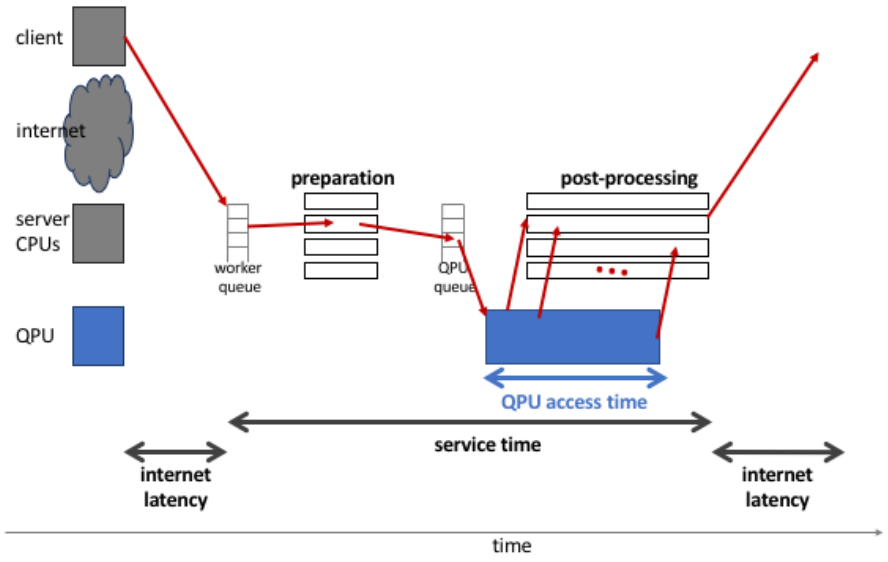


Figure 5.1: D-wave's operation and timing process. Available at [31]

.

The solver was run 20 times for each disruption to obtain a solution. The tables contained in this section present the average value of these iterations, and the full tables containing each iteration's values can be found in Appendix A. Table 5.3 presents the delay and swap penalties used in the QUBO, which were introduced in section 4.3. These values were chosen to equate the value of swapping two aircraft with the value of delaying a flight for one hour. In terms of dataset size, the QUBO model uses 30 flights starting from the first disrupted one and contains all the aircraft available.

δ	σ
0.01	0.3

Table 5.3: Delay and swap penalties values used in the program

5.3.1 First Experiment Result Analysis

Table 5.4 provides the results regarding solving five disruptions using Ferreira's [16] Particle Swarm Optimization (PSO) approach, which provided the overall quicker and better solutions

[1] https://docs.dwavesys.com/docs/latest/c_qpu_timing.html#how-solver-usage-is-charged

compared to the Ant Colony Optimization (ACO) algorithm, and this work's approach using D-wave's *LeapHybridSampler*. The time is regarding the total execution time and is expressed in milliseconds (ms). The delay is the solution's total delay for solving the disruption and is expressed in minutes (min).

Flight	**PSO [16]**		**Quantum**	
	Time (ms)	Delay (min)	Time (ms)	Delay (min)
928	168.9	0	40.6	0
1917	151.7	0	119.56	0
864	157.6	0	131.43	0
1614	87.7	0	97.91	5
839	59.7	0	95.36	80

Table 5.4: Results of the first experiment.

Figure 5.2 illustrates the comparison between the execution times of the PSO and the quantum implementations for a better visualization. Overall, it is possible to visualize that the timing is similar between both implementations, with some disruptions being faster in PSO and others in quantum. The main notable difference is regarding disruption 928, in which the PSO performed the slowest while the quantum approach provided the quickest execution time.

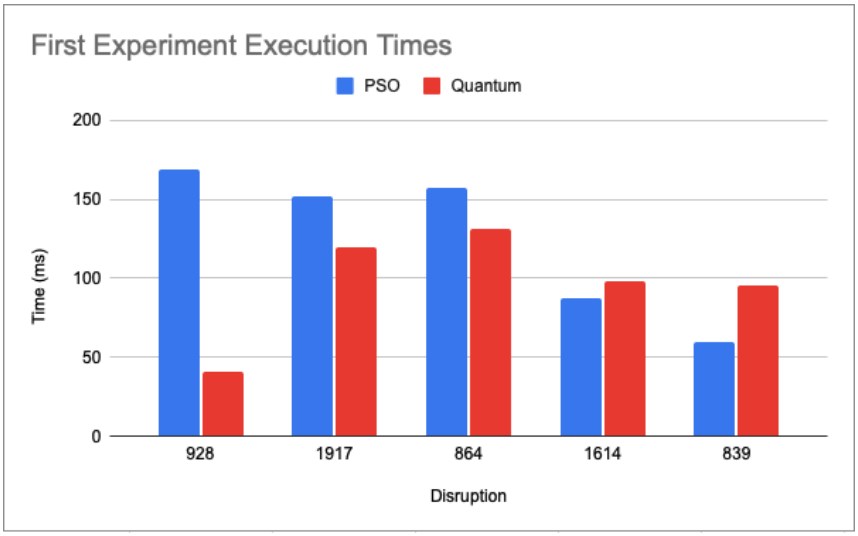


Figure 5.2: Execution times of the first experiment.

The quantum approach could provide valuable solutions in a feasible amount of time. The main difference between the results is visible in disruptions regarding flights 1614 and 839, where in this approach, they were delayed by 5 and 80 minutes, respectively. By analyzing both these flights, it is noticed that in the case of flight 1614, there is a delay of 5 minutes since the only other plane capable of performing it needs to delay the flight by 5 minutes to satisfy the minimum rotation time used in this work. In the case of flight 839, there is no other available aircraft in the airport that can carry the total number of passengers, which is a restriction adopted in this work.

As previously mentioned in section 3.2, TAP Air Portugal's central hub is located in Lisbon, where most of it departs or arrives. The number of TAP aircraft in Lisbon is more significant than in any other airport, especially considering other countries. This fact might explain why the quantum solver delayed the flights that originated from airports other than Lisbon, with Italy having the least alternatives to swap.

5.3.2 Second Experiment Result Analysis

Table 5.5 presents the results of resolving the second experiment. The table contains information regarding each disruption's affected resource, hard-change classification and the affected resource's fleet type. The table also provides data regarding each solver's solution, such as the execution time expressed in milliseconds, total delay in minutes, swaps and cancellations. The results of both solvers are very similar in terms of delays, swaps and cancellations, with their execution time being the main difference.

Regarding disruptions affecting flights 712, 686 and 152, considered hard-changes, the delay is the aggregation of all affected flights. In the case of hard-changes, delaying the flights is inevitable. Swapping aircraft does not have much influence due to their inability to depart the airport during the ongoing disruption. Disruptions affecting flights 139 and 916 are successfully solved by performing two swaps and avoiding delaying the disrupted flights, even in the case of flight 139 having a WB aircraft performing it.

Figure 5.3 illustrates a graph regarding the execution times for each solver during each disruption. It is possible to visualize that D-wave's *LeapHybridSolver* was significantly quicker than CPLEX in all cases except for disruption regarding flight 152, where CPLEX had the overall best timing in the experiment. Disruption affecting flight 152 is a hard-change regarding

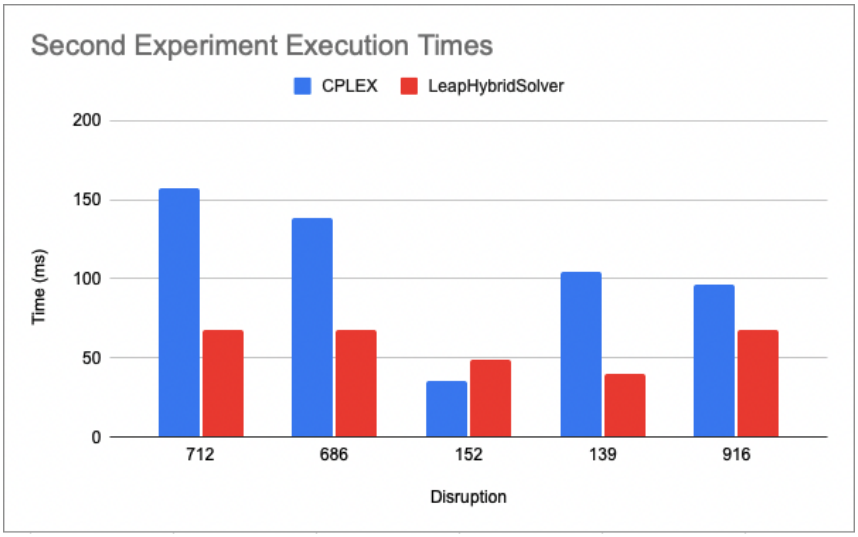


Figure 5.3: Execution times of the second experiment

Both solvers can resolve the problem quickly and provide feasible solutions while minimizing the impact on the flight plan. No cancellations were made in any disruptions, demonstrating that

the QUBO model was well constructed and prioritized swapping or delaying flights. Regarding execution time, D-wave's *LeapHybridSolver* managed to be quicker overall, but as mentioned at the beginning of the section, the time considered was regarding the QPU access.

5.4 Summary

This chapter presented the experiments conducted to evaluate the performance of the model introduced in chapter 4 to solve the Aircraft Recovery Problem (ARP). Two experiments were conducted, each using five different scenarios. The first experiment compared the model implemented in this work against Ferreira's [16] model, which used the same dataset. The second experiment used this model to compare classical and hybrid solvers in solving five disruptions. Due to the lack of metrics, in the first experiment, only the execution times and flight delays were compared, while in the second experiment, cancellations and swaps were also available. Regarding execution times with the hybrid solver, D-wave's QPU access time was considered, as the total execution time includes network delays and queues, as shown in figure 5.1. Overall, the model proposed in this work provided feasible solutions to minimizing delays, swaps and cancellations in a short execution time.

Disruption			**IBM's CPLEX**					**D-wave's LeapHybridSolver**				
Flight	**Hard Change**	**Fleet**	**Time (ms)**	**Delay (min)**	**Swaps**	**Canceled**	**Cost**	**Time (ms)**	**Delay (min)**	**Swaps**	**Canceled**	**Cost**
712	Yes	NB	157.54	94	0	0	0.94	67.93	94	0	0	0.94
686	Yes	NB	138.15	105	0	0	1.05	67.55	105	0	0	1.05
152	Yes	WB	35.61	90	0	0	0.9	48.98	90	0	0	0.9
139	No	WB	104.49	0	2	0	0.6	39.46	0	2	0	0.6
916	No	NB	96.55	0	2	0	0.6	67.62	0	2	0	0.6

Table 5.5: Results of the second experiment

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions of the work developed, challenges faced and possible future work. Section 6.1 provides an overview of what was accomplished in this dissertation. Section 6.2 explains the main difficulties faced during the development process. Section 6.3 presents the main contributions of this study in its field. Finally, section 6.4 suggests possibilities to increase the research and applicability of the Aircraft Recovery Problem (ARP) and quantum computing (QC) in the real world.

6.1 Conclusions

This work studied the Aircraft Recovery Problem (ARP), part of an airline's disruption management process. The ARP is a complex problem faced by airlines regularly that require a fast solution that minimizes the total cost and changes in the flight plan. The overall solution space of the problem is vast, with many alternatives such as delaying, swapping and canceling flights being considered in resolving the disruption. Due to recent impacts in the airline industry due to the pandemic, many airlines went bankrupt, and others are more financially pressured than ever. Therefore, an efficient and quick way to recover from disruptions, which unavoidably happen, is necessary to avoid more financial losses.

The field of quantum computing (QC) was also studied in this work. QC is a field that combines computing with quantum mechanics, intending to solve complex problems faster than classical computers. Optimization problems are commonly modeled as a Quadratic Unconstrained Binary Optimization (QUBO) model when solving using QC. Companies have built quantum computers capable of solving this type of problem and also have libraries for modeling them. Given that the ARP is a very complex optimization problem and the premise of QC being faster than classical computers, a Two-Stage algorithm to model the problem into a QUBO to be solved in a quantum computer was developed and presented in chapter 4.

In chapter 5, two experiments were conducted to evaluate the proposed model's performance when solved using QC, each with five different scenarios. The first experiment compared Ferreira's [16] approach to the one developed in this work since the same dataset was used in both works. The first experiment's only available metrics for comparison were execution times and flight delays. Overall, the solutions were very similar in execution times and results, with the main differences resulting from the different constraints considered. The second experiment compared the performance of a classical solver against a hybrid solver using the model implemented in this work, with the availability of other metrics such as swaps and cancellations. The solutions were the same for every scenario, with the main difference being the execution time, which was faster using the hybrid solver.

Overall, the model implemented in this work successfully solved the ARP quickly while minimizing the total delays, swaps and cancellations. Still, as mentioned in chapter 5, using a quantum computer was not an option due to qubit and time limitations which would require a drastic reduction of the problem's complexity, so a hybrid option was used instead. In most cases, the results showed that using a hybrid solver may reduce the execution time of obtaining a solution compared to a classical solver. Due to the performance of the hybrid solver and regarding the premises of quantum computers, a quantum solver might achieve a significant reduction in execution times when solving the ARP.

6.2 Main Difficulties

The development process of this work faced many different challenges. The ARP is a complex problem faced by airlines every day and involves many different factors while also requiring a quick and low-cost solution. It is a field of constant research with authors proposing different implementations to tackle the problem efficiently. However, the overall complexity of the ARP and the requirement of providing a quick, low-cost solution is still a problem. Implementing the ARP as a Quadratic Unconstrained Binary Optimization (QUBO) model also provided difficulties since many different constraints had to be considered and added to the model. Using binary variables assigning an aircraft to a flight required the modeling process to reduce the number of variables to reduce the overall complexity of the problem.

QC is still an emerging field in computation based on quantum mechanics, which requires much study in the area to understand its benefits and how it functions. Quantum computers are currently very limited in available qubits, which may require a reduction of the complexity of the problem to be able to solve. Regarding availability, few companies currently build and offer quantum computers; most require a subscription to use the computers without time or memory limitations.

6.3 Main Contributions

The goal of this dissertation was to use quantum computing to solve the Aircraft Recovery Problem (ARP). The implementation of the model presented in chapter 4 brought significant contributions to the fields of quantum computing (QC) and airline disruption management.

The modeling of the problem into a Quadratic Unconstrained Binary Optimization (QUBO) model contributes to the emerging field of QC, especially regarding optimization problems. As previously mentioned, the QUBO model is a popular option for modeling problems to be solved using QC. However, the field is still emerging, and the model implemented in this work may contribute to future developments.

Regarding the ARP, to the best of the author's knowledge, this is the first approach to solving the ARP using QC. Some studies solve other airline industry problems using QC, such as the Tail Assignment Problem, but this is the first one regarding the disruption management process. This development may contribute to further research in the airline industry regarding the usage of quantum computers to solve optimization problems that require a quick execution time with low-cost solutions.

6.4 Future Work

This work covered the modeling of the ARP into a QUBO to be solved using quantum computing. As presented in section 4.1.3, the decision to model the problem as a QUBO was made based on the availability of libraries for modeling such problems by companies that offer quantum computers capable of solving them. Another contributing factor was the availability of literature regarding other optimization problems solved the same way.

A critical alternative would be trying the QUBO implementation using a different dataset to verify its applicability in more scenarios, especially when dealing with airlines composed of a different fleet. Other tasks, such as mandatory aircraft maintenance, could be included in the model due to its requirement in real-life. The availability of more metrics to compare the model to would also be crucial. Instead of having a binary variable representing the assignment of an aircraft to a flight, one could represent the assignment of an aircraft to a sequence of flights, aiming to reduce the number of variables in the problem.

Another possible future work would be adding even more constraints to the problem, since airline's have more restrictions in the assignment of aircraft and flights. The addition of scheduled maintenance tasks to the model should be an important integration, since the aircraft to be maintained needs to be in the correct airport during that time and is a realistic task performed constantly.

References

[1] Airports Council International (ACI). Passenger protection under cases of flight disruption. Technical report, WORLDWIDE AIR TRANSPORT CONFERENCE (ATCONF), 3 2013.

[2] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, and Chun-Fu Chen. Qiskit: An open-source framework for quantum computing, 2019.

[3] AG Aruna, KH Vani, and RS Meena. A study on reversible logic gates of quantum computing. *International Journal of Computer Science and Information Technologies*, 7(1):427–432, 2016.

[4] Serge Bisaillon, Jean-François Cordeau, Gilbert Laporte, and Federico Pasin. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR*, 9:139–157, 2011.

[5] Amira Bouaziz, Amer Draa, Salim Chikhi, and Algeria Constantine. A quantum-inspired artificial bee colony algorithm for numerical optimisation. *2013 11th International Symposium on Programming and Systems (ISPS)*, 2013.

[6] D L Bryan and Morton O'Kelly. Hub-and-spoke networks in air transportation: An analytical review. *Journal of Regional Science*, 39:275–295, 6 1999.

[7] António J. M. Castro, Ana Rocha, and Eugénio Oliveira. A new approach for disruption management in airline operations control. *AI Matters*, 1:15–16, 08 2014.

[8] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry—concepts, models and methods. *Computers & Operations Research*, 37:809–821, 5 2010.

[9] Jens Clausen, Jesper Larsen, Allan Larsen, and Jesper Hansen. Disruption management-operations research between planning and execution. *Or/ms Today*, 28:40–43, 2001.

[10] David Elieser Deutsch and Roger Penrose. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425:73–90, 1989.

[11] P. A.M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35:416–418, 1939.

[12] Kaitano Dube, Godwell Nhamo, and David Chikodzi. Covid-19 pandemic and prospects for recovery of the global aviation industry. *Journal of Air Transport Management*, 92:102022, 2021.

[13] Maximilian Etschmaier and Dennis Mathaisel. Airline scheduling: An overview. *Transportation Science*, 19:127–138, 2 1985.

[14] EUROCONTROL. All-causes delay and cancellations to air transport in europe - 2019. Technical report, EUROCONTROL, 2020.

[15] Jan Evler, Martin Lindner, Hartmut Fricke, and Michael Schultz. Integration of turnaround and aircraft recovery to mitigate delay propagation in airline networks. *Computers & Operations Research*, 138:105602, 2022.

[16] António José Ferreira. Evolutionary computation methods applied to operational control centers. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2015.

[17] Fred W. Glover and Gary A. Kochenberger. A tutorial on formulating QUBO models. *CoRR*, abs/1811.11538, 2018.

[18] Tobias Grosche. *Airline Scheduling Process*. Springer Berlin Heidelberg, 2009.

[19] Yourui Huang, Chaoli Tang, and Shuang Wang. Quantum-inspired swarm evolution algorithm. *Proceedings - CIS Workshops 2007, 2007 International Conference on Computational Intelligence and Security Workshops*, pages 208–211, 2007.

[20] International Civil Aviation Organization (ICAO). Manual on the regulation of international air transport. Technical report, International Civil Aviation Organization (ICAO), 2004.

[21] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28:58–81, 2014.

[22] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13:149–162, 5 2007.

[23] Luis N. Martins, Ana Paula Rocha, and Antonio J.M. Castro. A qubo model to the tail assignment problem. *ICAART 2021 - Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 2:899–906, 2021.

[24] Hamed Mohammadbagherpoor, Patrick Dreher, Mohannad Ibrahim, Young-Hyun Oh, James Hall, Richard E Stone, and Mirela Stojkovic. Exploring airline gate-scheduling optimization using quantum computers, 2021.

[25] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[26] S.E. Rasmussen, K.S. Christensen, S.P. Pedersen, L.B. Kristensen, T. Bækkegaard, N.J.S. Loft, and N.T. Zinner. Superconducting circuit companion—an introduction with worked examples. *PRX Quantum*, 2(4), Dec 2021.

[27] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Comput. Surv.*, 32(3):300–335, sep 2000.

[28] Eleanor Rieffel and Wolfgang Polak. *Quantum Computing: A Gentle Introduction*. The MIT Press, 1st edition, 2011.

[29] Kim Riis, Larsen Jesper, Clausen Jens Løve Michael, and Sørensen. Disruption management for an airline — rescheduling of aircraft. *Applications of Evolutionary Computing*, pages 315–324, 2002.

[30] Benjamin Schumacher. Quantum coding. *Phys. Rev. A*, 51:2738–2747, Apr 1995.

[31] D-Wave Systems. Operation and timing — d-wave system documentation. https://docs.dwavesys.com/docs/latest/_images/highlevel_timing.png. Accessed 2022-07-20.

[32] J. Vink, B.F. Santos, W.J.C. Verhagen, I. Medeiros, and R. Filho. Dynamic aircraft recovery problem - an operational decision support framework. *Computers & Operations Research*, 117:104892, 5 2020.

[33] Ling Wang, Qun Niu, and Minrui Fei. A novel quantum ant colony optimization algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4688 LNCS:277–286, 2007.

[34] Seyed Hessameddin Zegordi and Niloofar Jafari. Solving the airline recovery problem by using ant colony optimization. *International Journal of Industrial Engineering & Production Research*, 21:121–128, 09 2010.

[35] Cheng Zhang. Two-stage heuristic algorithm for aircraft recovery problem. *Discrete Dynamics in Nature and Society*, 2017, 2017.

[36] Ai Zhao, Jonathan F. Bard, and J. Eric Bickel. A two-stage approach to aircraft recovery under uncertainty. *SSRN Electronic Journal*, 2022.

Appendix A

Experiments Execution Times

This appendix contains all the execution times and solutions measured for each experiment presented in chapter 4. Section A.1 presents the data for the first experiment and section A.2 presents the data for the second experiment.

A.1 First Experiment Results

This section presents the execution times for obtaining the first experiment results, previously mentioned in section 5.3.1. Table A.1 contains the data regarding D-wave's *LeapHybridSolver* for solving the five disruptions, and contains the execution time and delay for each iteration run.

A.2 Second Experiment Results

This section provides the total executions performed for obtaining the second experiment results, presented in section 5.3.2. Section A.2.2 provides the tables regarding D-wave's *LeapHybridSolver* results and section A.2.1 regarding IBM's CPLEX results.

A.2.1 CPLEX Results

The two tables presented in this section contain the results of twenty iterations including each solutions execution time, delay, swaps and cancellations. Table A.2 contains the data for disruptions affecting flights 712, 686 and 152, and table A.3 for disruptions affecting flights 139 and 916.

A.2.2 D-wave's LeapHybridSolver Results

The two tables presented in this section contain the results of twenty iterations including each solutions execution time, delay, swaps and cancellations. Table A.4 contains the data for disruptions affecting flights 712, 686 and 152, and table A.5 for disruptions affecting flights 139 and 916.

Execution Times and Delays for disruptions 928, 1917, 864, 1614 and 839										
Iteration	**928**		**1917**		**864**		**1614**		**839**	
	Time	**Delay**	**Time**	**Delay**	**Time**	**Delay**	**Time**	**Delay**	**Time**	**Delay**
1	43.0	0	128.01	0	132.42	0	107.2	5	95.41	80
2	40.73	0	170.2	0	124	0	104.1	5	93.59	80
3	45.05	0	94.01	0	125.8	0	119.35	5	95.05	80
4	39.21	0	131.2	0	102.5	0	101.77	5	97.83	80
5	39.66	0	108.1	0	126.77	0	98.81	5	140.31	80
6	33.26	0	132.92	0	103.93	0	117.91	5	69.79	80
7	38.22	0	127.6	0	97.22	0	93.86	5	92.2	80
8	39.8	0	109.1	0	130.18	0	110.55	5	97.16	80
9	47.23	0	82.68	0	128.61	0	79.88	5	94.76	80
10	37.35	0	148.88	0	132.78	0	113.93	5	85.94	80
11	43.13	0	131.52	0	153.04	0	90.98	5	105.68	80
12	42.56	0	99.76	0	140.24	0	106.44	5	102.93	80
13	33.32	0	95.58	0	126.53	0	85.98	5	94.61	80
14	37.37	0	118.48	0	162.22	0	91.82	5	80.66	80
15	38.93	0	108.17	0	135.15	0	77.76	5	86.1	80
16	45.78	0	100.11	0	129.1	0	115.98	5	114.73	80
17	45.58	0	115.54	0	144.69	0	69.34	5	103.49	80
18	39.11	0	115.16	0	138.66	0	101.75	5	88.32	80
19	39.67	0	132.41	0	151.48	0	72.6	5	72.91	80
20	43.56	0	141.79	0	143.28	0	98.21	5	95.73	80

Table A.1: First experiment results. Time is expressed in milliseconds and delay in minutes

Execution Times, delays, swaps and cancellations for disruptions 712, 686 and 152 using IBM's CPLEX												
Iteration	**712**				**686**				**152**			
	Time	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**
1	157.5	0	0	0	135.44	0	0	0	39.93	0	0	0
2	169.61	0	0	0	137.62	0	0	0	37.66	0	0	0
3	154.4	0	0	0	137.7	0	0	0	38.0	0	0	0
4	154.8	0	0	0	135.46	0	0	0	35.97	0	0	0
5	158.02	0	0	0	139.68	0	0	0	36.85	0	0	0
6	161.32	0	0	0	134.66	0	0	0	37.56	0	0	0
7	169.61	0	0	0	140.06	0	0	0	37.2	0	0	0
8	168.11	0	0	0	138.3	0	0	0	34.5	0	0	0
9	156.25	0	0	0	134.85	0	0	0	36.06	0	0	0
10	159.99	0	0	0	143.57	0	0	0	36.2	0	0	0
11	168.96	0	0	0	134.41	0	0	0	35.03	0	0	0
12	150.66	0	0	0	139.64	0	0	0	34.49	0	0	0
13	158.1	0	0	0	138.63	0	0	0	31.06	0	0	0
14	156.4	0	0	0	141.8	0	0	0	32.5	0	0	0
15	176.05	0	0	0	133.91	0	0	0	33.66	0	0	0
16	135.43	0	0	0	138.48	0	0	0	33.99	0	0	0
17	162.03	0	0	0	137.62	0	0	0	35.48	0	0	0
18	141.61	0	0	0	146.23	0	0	0	33.41	0	0	0
19	149.73	0	0	0	140.31	0	0	0	35.45	0	0	0
20	142.22	0	0	0	134.63	0	0	0	37.21	0	0	0

Table A.2: Second experiment results for IBM's CPLEX regarding disruptions on flights 712, 686 and 152. Time is expressed in milliseconds and delay in minutes

Execution Times, delays, swaps and cancellations for disruptions 129 and 916 using IBM's CPLEX								
Iteration	**139**				**916**			
	Time	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**
1	104.89	0	2	0	97.95	0	2	0
2	105.8	0	2	0	99.7	0	2	0
3	116.67	0	2	0	95.42	0	2	0
4	109.4	0	2	0	99.52	0	2	0
5	99.6	0	2	0	106.48	0	2	0
6	106.32	0	2	0	95.73	0	2	0
7	100.58	0	2	0	96.8	0	2	0
8	96.09	0	2	0	92.43	0	2	0
9	93.78	0	2	0	91.19	0	2	0
10	104.21	0	2	0	96.26	0	2	0
11	122.19	0	2	0	92.03	0	2	0
12	97.86	0	2	0	98.53	0	2	0
13	106.7	0	2	0	95.09	0	2	0
14	114.94	0	2	0	98.87	0	2	0
15	101.39	0	2	0	95.56	0	2	0
16	98.77	0	2	0	92.08	0	2	0
17	98.27	0	2	0	96.6	0	2	0
18	111.75	0	2	0	96.44	0	2	0
19	96.51	0	2	0	96.88	0	2	0
20	104.08	0	2	0	97.44	0	2	0

Table A.3: Second experiment results for IBM's CPLEX regarding disruptions on flights 129 and 916. Time is expressed in milliseconds and delay in minutes

Execution Times, delays, swaps and cancellations for disruptions 712, 686 and 152 using D-wave's LeapHybridSolver												
Iteration	**712**				**686**				**152**			
	Time	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**
1	63.8	94	0	0	75.98	105	0	0	56.45	90	0	0
2	55.35	94	0	0	74.9	105	0	0	52.9	90	0	0
3	67.58	94	0	0	81.82	105	0	0	52.29	90	0	0
4	54.58	94	0	0	51.58	105	0	0	40.37	90	0	0
5	77.34	94	0	0	52.79	105	0	0	48.66	90	0	0
6	70.6	94	0	0	55.16	105	0	0	46.23	90	0	0
7	61.27	94	0	0	81.01	105	0	0	35.84	90	0	0
8	68.68	94	0	0	41.67	105	0	0	48.84	90	0	0
9	86.86	94	0	0	56.9	105	0	0	60.64	90	0	0
10	65.25	94	0	0	69.54	105	0	0	52.44	90	0	0
11	64.14	94	0	0	64.87	105	0	0	47.23	90	0	0
12	83.67	94	0	0	92.33	105	0	0	55.33	90	0	0
13	59.24	94	0	0	89.69	105	0	0	38.26	90	0	0
14	64.78	94	0	0	60.39	105	0	0	42.87	90	0	0
15	76.79	94	0	0	97.63	105	0	0	39.13	90	0	0
16	79.86	94	0	0	45.5	105	0	0	52.7	90	0	0
17	72.96	94	0	0	70.33	105	0	0	49.32	90	0	0
18	61.77	94	0	0	83.91	105	0	0	56.18	90	0	0
19	55.34	94	0	0	55.5	105	0	0	46.93	90	0	0
20	68.74	94	0	0	49.5	105	0	0	56.99	90	0	0

Table A.4: Second experiment results for D-wave's LeapHybridSolver regarding disruptions on flights 712, 686 and 152. Time is expressed in milliseconds and delay in minutes

Execution Times, delays, swaps and cancellations for disruptions 129 and 916 using D-wave's LeapHybridSolver								
Iteration	**139**				**916**			
	Time	**Delay**	**Swaps**	**Cancellations**	**Time**	**Delay**	**Swaps**	**Cancellations**
1	41.86	0	2	0	57.92	0	2	0
2	31.42	0	2	0	72.55	0	2	0
3	36.83	0	2	0	56.86	0	2	0
4	40.05	0	2	0	62.7	0	2	0
5	41.32	0	2	0	76.17	0	2	0
6	40.59	0	2	0	57.2	0	2	0
7	42.71	0	2	0	77.34	0	2	0
8	42.46	0	2	0	78.68	0	2	0
9	41.74	0	2	0	83.76	0	2	0
10	39.55	0	2	0	59.82	0	2	0
11	37.49	0	2	0	54.59	0	2	0
12	36.71	0	2	0	69.32	0	2	0
13	38.07	0	2	0	66.57	0	2	0
14	41.24	0	2	0	79.02	0	2	0
15	37.83	0	2	0	75.38	0	2	0
16	36.51	0	2	0	58.85	0	2	0
17	36.84	0	2	0	58.72	0	2	0
18	41.08	0	2	0	65.2	0	2	0
19	42.55	0	2	0	78.63	0	2	0
20	42.35	0	2	0	63.12	0	2	0

Table A.5: Second experiment results for D-wave's LeapHybridSolver regarding disruptions for flights 129 and 916. Time is expressed in milliseconds and delay in minutes