Passenger Re-accommodation for a Planned Schedule Change

A Project Report Submitted
in Fulfillment of the Requirements
for Problem Statement at

12th Inter IIT Tech Meet

by

Team 79

to the

Mphasis Ltd.

Contents

Chapter 1

Introduction

In this project, we have attempted to solve a common problem faced by airline companies, which is the Passenger Re-accommodation for a Scheduled Flight Change.

1.1 Problem Definition

Oftentimes, the schedules of flights get changed due to some unforeseeable circumstances and the flights get delayed/cancelled. In such a situation, the passengers of the affected flight would have to be transferred to any feasible flight which ultimately leads to their destination.
However, this process is not completely straightforward as there are many constraints to be considered such as -

- Finding suitable flights among a myriad of flights throughout the country
- Ensure the considered flights have convenient timing for a smooth transfer
- Ensuring all of the passengers have the required free seats and inventory
- Assigning all of the passengers to their new flights based on their current seat choice or providing the next best option of seat change for their paid money

Given the unpredictability of these problems, they pop up at the hands of the airline management at very short notice; thus swift action is of utmost importance.
In this project, we have given a Quantum solution to this problem which guarantees a fast and correct flight path charting and passenger re-accommodation.

1.2 Introduction to Quantum Computing

Quantum Computing bases its computations on the principles of Quantum Mechanics.
In everyday computational methods (aka Classical Computing), the fundamental unit of computation is a **bit** which can have two values of 0/1.
In Quantum Computing, we deal with a **Qubit**, which in succinct terms, can be thought of as a particle with two energy levels (or states). Mathematically, it can be represented

by a 2-dimensional complex vector.

Some fascinating concepts or properties in Quantum Computing are:

- By the laws of Quantum Mechanics, a qubit can be in the *superposition* of both states.
- It can also be *entangled* with another qubit where both qubits' states are correlated.
- By the famous Schrodinger's Paradox involving a cat, it is seen that when we observe a system (or *measure* it), the system in a possibly unknown state collapses to one known state.
- The state of a qubit can be changed by performing certain operations which can be mathematically represented by matrices.

Using the above principles, we can play around with the quantum systems to craft a plethora of Quantum Algorithms for our usage. Due to the innate parallelism of Superposition and Entanglement, some of these algorithms can perform better than their classical counterparts.

Chapter 2

Identification

This part of our works takes the data set as input and gives the following output :

1. A python dictionary with keys as Inventory ID's of the impacted flights and values as the type of change for that impacted flight.
2. A database table for each impacted flight with table name being the Inventory ID of the impacted flight and rows in table being the PNR Booking entries of that impacted flight.

This is done in 3 steps the details of which are given belo.

2.1 Pre processing

The data set we were given is in csv format and we converted it into a python database so that searching for impacted flights and passengers would become easier.

2.2 Assumptions

The following minimal assumptions we assumed about the planned schedule changes :

1. If flight is cancelled then the flight number and the date for which it is cancelled.
2. If the flight is delayed then the new departure date and time, the new arrival date and time and also the flight number and the date for which it is cancelled.

2.3 Identifying the impacted flights and passengers

To get the inventory ID of the impacted flight and the impacted passengers for that flight SQL queries are used.

Chapter 3

Optimisation

This part of the Project involves some concepts of Quantum Computing.
Our Optimisation is based on modelling the given problem into a Quadratic Program and finding an optimal solution through QAOA (Quantum Approximate Optimisation Algorithm).

3.1 Quadratic Unconstrained Binary Optimisation (QUBO)

In Quadratic programming (QP), one seeks to optimise a multivariate quadratic function subject to linear constraints on the variables.

3.1.1 Quadratic Program

A Quadratic program problem with N variables and M constraints can be formulated as follows:

1. a real-valued, N-dimensional vector c
2. an $N \times N$-dimensional real symmetric matrix Q
3. an $M \times N$-dimensional real matrix A
4. an M-dimensional real vector b

The Objective of the program is to find an N-dimensional vector $\mathbf{x}$ such that:

$$\text{minimize } \mathbf{x}^{\mathrm{T}} Q \mathbf{x} + \mathbf{c}^{\mathrm{T}} \mathbf{x}$$

$$\text{subject to } A\mathbf{x} \preceq \mathbf{b}$$

3.1.2 QUBO

In QUBO, we deal with equations with no constraints. So the problem resolves to the following form:

Given a real-valued upper triangular matrix $Q \in R^{N\times N}$, whose entries Q_{ij} define a weight for each pair of indices $i, j \in \{1, \dots, N\}$ within the binary vector.
We define a function $f_Q : \{0,1\}^N \to R$ as:

$$f_Q(x) = x^\top Q x = \sum_{i=1}^{n}\sum_{j=i}^{n} Q_{ij} x_i x_j$$

Thus, The QUBO problem is defined as:

$$x^* = \underset{x\in\{0,1\}^N}{\arg\min}\, f_Q(x)$$

The QUBO Approach is used in a lot of combinatorial problems such as MaxCut, Graph coloring, Graph partitioning,etc.
QUBO is a NP-Hard problem but it can be relaxed into easier problems. We make use of such relaxations.

3.2 Quantum Approximate Optimisation Algorithm (QAOA)

This is a hybrid algorithm involving both quantum and classical operators.
It is a general technique that can be used to find approximate solutions to combinatorial optimisation problems which can be cast as searching for an optimal bitstring.

It involves a back-and-fro interaction between the Quantum Backend and the Classical Optimiser.

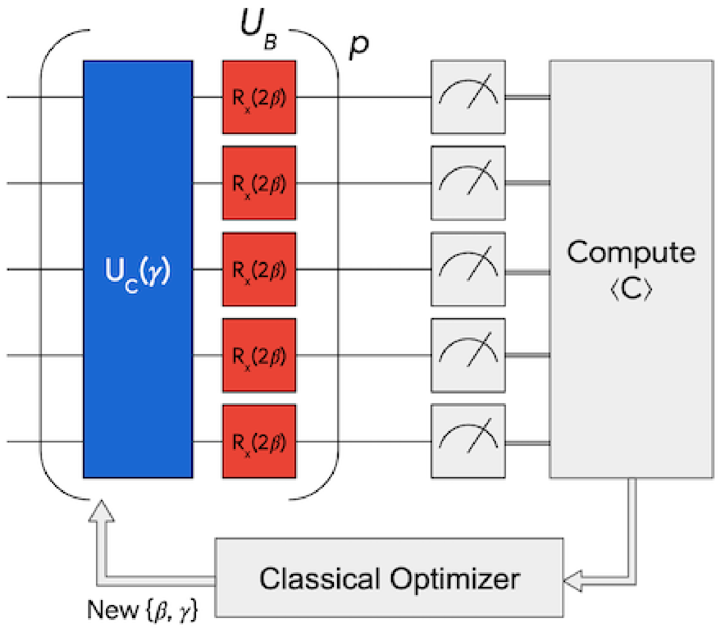

Fig. 3.1 QAOA Schema, credits: Cirq Google QuantumAI

It is a type of a variational Quantum Algorithm whose general schema is as follows:

1. The quantum backend is initialised with an initial state.

2. A parameterised Ansatz circuit is set up

3. We evaluate the cost function (or the expectation value of the energy of the system) and provide the results to the Classical Optimiser

4. The Classical Optimiser will update the parameter values of the Circuit

5. The new parameter values are fed into the quantum backend which runs again

The starting point of QAOA is the specification of cost and mixer operators (known as Hamiltonians). These Hamiltonians are used to alter the state of the qubits (or equivalently the energy of the system), according to the varying parameters, guiding the system to a target state (or a target energy configuration). We use time evolution and layering to create a variational circuit and optimise its parameters. The algorithm concludes by sampling from the circuit to get an approximate solution to the optimisation problem.

The Quantum Processing of QAOA consists of the following steps:

1. Define a cost Hamiltonian H_C such that its ground state encodes the solution to the optimisation problem.

2. Define a mixer Hamiltonian H_M, which changes the probability distribution of the state (which can be seen as changing the energy of the system to allow it to attain a minimum).

3. Construct the cost layer ($e^{-i\gamma H_C}$) and mixer layer ($e^{-i\alpha H_M}$).

4. Choose a parameter $n \geq 1$ and build the circuit consisting of repeated application of the cost and mixer layers ($U(\gamma, \alpha) = e^{-i\alpha_n H_M} e^{-i\gamma_n H_C} ... e^{-i\alpha_1 H_M} e^{-i\gamma_1 H_C}$).

5. Prepare an initial state, apply $U(\gamma, \alpha)$, and use classical techniques to optimize the parameters.

6. After the circuit has been optimised, measurements of the output state reveal approximate solutions to the optimisation problem.

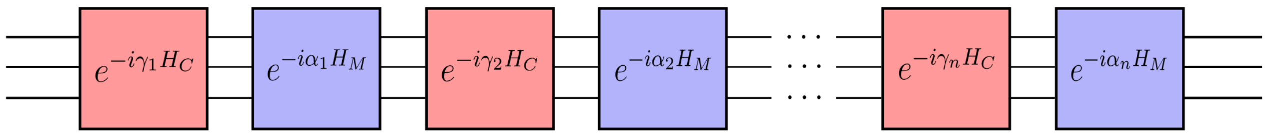

Fig. 3.2 QAOA Parameterized Circuit, credits: PennyLane

By itself, the cost Hamiltonian cannot change the energy of the state and hence cannot lead to any optimisation. By allowing for a mixer Hamiltonian, we do not conserve the energy of the system and allow it to be changed to reach a minimum.

Here, each of the binary variables corresponds to a qubit.
The initial state is a parameterized circuit consisting of p unknowns.

3.2.1 Warm Start QAOA

It is possible to increase the efficiency of the QAOA by choosing an appropriate initial state. Here we relax the problem to have continuous variables rather than binary variables. So we start with an initial solution to the continuous relaxed version of the QUBO such as:

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} R_y(2 \arcsin \sqrt{c_i^*})|0\rangle_n$$

Here c_i^* is the value of the variable x_i of the relaxed problem.

By specifying a better initial state, we have decreased the number of iterations required to reach the optima.

3.3 Our Procedure

Input -

1. A dictionary of inventory IDs with the flight status of being cancelled or delayed. (we are using inventory IDs as it is unique to each day's flight)
2. Inventory Dataframe
3. Inventory Id

Output - Paths - Dictionary with keys as affected flight inventory ids and their alternate flight paths

3.3.1 Our Approach

We map all the flights as binary variables: whether they are part of the final flight paths. Let n be the total number of flights in consideration. Thus we construct the Quadratic Program as follows:
We would consider the following penalties into account :

1. Q Matrix: Measures the total time taken by the flight (Arr - Dep of the flight)
2. A Matrix: Checks if the current flight departs from the same airport as the Affected flight
3. B Matrix: Checks if the current flight arrives at the same airport as the Affected flight

4. N Matrix: Checks the neighbouring flight connectivity (by departure)

5. G Matrix: Measures the ground time spent by a flight (next departure - arrival)

6. D Matrix: Measures the total delay of a path

This forms a Quadratic Program of the form:

$$\min_{x_i; i \in \{0..n\}} xQx^T + xGx^T + Dx$$

subject to :

$$1 \leq \text{Ix} \leq 5$$

$$\text{Ax} = 1$$

$$\text{Bx} = 1$$

$$\forall i \in \{0..n\} \ N_i x \geq 1$$

We then convert this into a QUBO (Quadratic Unconstrained Binary Optimisation) instance and feed it into the QAOAsolver.

3.3.2 Pre-processing

Before we convert into a Quadratic Program, we do an extensive pre-processing of the data, to filter out irrelevant flights.
We set up some rules to not consider some of the flights:

1. Do not consider flights which are cancelled.

2. Do not consider flights which arrive at the affected flight's departure airport and vice versa.

3. If the affected flight is delayed, then do not consider flights arriving later than the affected flight

4. Do not consider the flights which violate the Minimum Ground Time and Maximum Connecting Time

5. Do not consider the flight if its flight window is more than that of the affected flight, and it departs and arrives at different airports than of the affected flights

6. Remove the flights with no predecessor or successor.

3.3.3 Quantum Solver

Once we have the list of feasible flights and quadratic program created, we start the Quantum Optimisation process.

Here we use two Classical Optimisers:

1. COBYLA - Constrained Optimization by Linear Approximation
2. Cplex - IBM Optimisation Software Package

A brief schema is as follows:

1. We create a QAOA instance of the Quadratic program, using COBYLA optimiser
2. We feed the QAOA instance into the WarmStartQAOAOptimiser which takes a CplexOptimiser and creates a WarmStart instance of the QAOA object.
3. We get the solution of the Quadratic Program through the WarmStartOptimiser

3.3.4 Post Processing

The output from the Quantum Solver is a bit string. Each character in the bitstring signifies whether a particular flight was considered in the final flight path.
The QuantumSolver will yield the optimal flight path for a given flight. As we are dealing with passenger re-accommodation, we would need other sub-optimal flights too. So we increase the penalty on the best optimal path to get other flight paths.

3.3.5 Alternate Methods

Alternate methods we looked at -

1. Grover Search:
    - Quantum Search Algorithm faster than classical algorithms searching through an unstructured database
    - Assumptions: No of solutions needed to be known prior
2. Quantum Walk:
    - Walk through a graph using a coin operator, which allows walking through multiple nodes simultaneously
    - Not guaranteed to return a path for a general graph

While exploring different algorithms, we considered the quantum walk due to its ability to walk through multiple nodes simultaneously. Each airport, a node, was mapped to a number. We encoded the qubits such that each K+B qubit represents one node in a path. We used K qubits to represent the location of the node among the children of its parent. The next B qubits were used to store the value of the node. So, the algorithm would return a path as:

$$\text{Path}=\underbrace{\{K+B\}}_{\text{node 1}},\underbrace{\{K+B\}}_{\text{node 2}},.....\underbrace{\{K+B\}}_{\text{node n}}$$

We discarded this method as it went exponential when we tried to assign the value of the node to the qubits.

Chapter 4

Allocation

This section involves the allocation of impacted passengers to alternate flights based on the quantum solution. In post-processing, the quantum QUBO solution for flight reallocation and passenger class adjustments are the key steps involved. The process is structured to efficiently handle inventory identification, passenger prioritization, class adjustments, and passenger allocation ensuring a smooth transition for affected passengers.

4.1 Inventory Identification and Reallocation

1. Utilizing the pre-processing, we systematically identify the inventory IDs of affected passengers requiring flight reallocation. This involves an assessment of alternative flights to accommodate passengers with disrupted itineraries.
2. The identified flights were carefully chosen to ensure seat availability, guaranteeing a seamless transition for affected passengers to alternate flights.

4.2 Passenger Prioritization and Score Assignment

We assign a unique score to each passenger by incorporating passenger details from the PNR booking dataset, including special requests, age, and cabin class. This score is calculated using a rule-based engine, prioritizing passengers based on their specific requirements and preferences. The organization of the PNR dataset by passenger scores ensures a systematic and fair allocation of seats on alternate flights, giving higher precedence to passengers with higher scores.

4.3 Class Upgradation and Downgradation

With comprehensive knowledge of both flights and passengers, a critical analysis is conducted to assess class upgradation or downgradation possibilities. This step is driven by passengers' preferences and available seat capacities in various cabin classes. Passengers are considered for class adjustments based on their preferences, enabling us to optimize seat utilization while meeting passengers' expectations.

4.4 Allocation of Seats

1. The final phase involves the allocation of the available seats, aligning with the determined class and flight number. This allocation process ensures optimal utilization of available seats and minimises disruptions for passengers with altered itineraries.

2. Reallocation is performed to maximise passenger satisfaction while maintaining operational efficiency.

Chapter 5

Conclusion

The team extracted information corresponding to the affected flights and gave the passengers a list of cancelled/delayed flights. The set of alternate flights to substitute the cancelled/delayed flights was obtained through the quantum algorithms mentioned above. The affected passengers were then sorted per their priorities and allocated alternate flight solutions.
We have also provided a feature of mailing the affected passengers about the current status of the flight and their available options.

5.1 Performance Superiority of QAOA over Classical Algorithms

The team made use of quantum algorithms to make exponential classical solutions more efficient. To that end, the QAOA(Quantum Approximate Optimisation Algorithm) is exponentially better than the classical solution in space complexity at the same time offering marginal improvements over the classical solution in time complexity, as QAOA is a heuristic algorithm.

Chapter 6

References

1. IBM Qiskit used to code the Quantum Optimisation
2. Replanning Flight Schedules using Quantum Computing by André Mamprin Mori