Mphasis

The Next Applied

Quantum Computing Challenge

Inter IIT Tech Meet 12.0

Final Submission Report

Author:

Team 15

Contents

1 Demonstrative Video **2**

2 Introduction **2**

3 Background Literature review **2**

4 Problem Statement **2**
4.1 Problem Formulation . . . 2
4.2 List of Flight Disruptions . . . 3
4.3 Requirements . . . 3
4.4 Business Challenges . . . 3

5 Solution Methodology **4**
5.1 Identification of Impacted PNRs . . . 4
5.2 Development of the Airline Route Graph . . . 6
5.3 Identification of Alternate Feasible Flights . . . 6
5.4 Cost Function . . . 7
5.5 CQM Modelling of the Problem . . . 9
5.5.1 Objective Function . . . 9
5.5.2 Constraints . . . 10
5.6 Quantum Strategy using D-Wave . . . 11
5.7 Consideration of different City Pairs . . . 11
5.8 Quantum Network Flow . . . 12
5.9 Customer Feedback . . . 14

6 Flexibility **14**

7 Results **15**

8 Discussion and Scalability **16**

9 Conclusion **17**

A D-Wave Quantum Computers **18**

B BQM vs CQM **19**

C Pictures of the Business Rule Engine **20**

1 Demonstrative Video

Kindly find the demonstrative video of our solution to the problem statement, the proposed business rule engine here.

2 Introduction

In the dynamic and ever-evolving airline industry, flight cancellations and schedule changes are a common occurrence. These changes can be driven by multiple factors such as seasonal demands, bad weather, introduction of new routes, adjustments for daylight savings, etc. Such alterations, while necessary for operational efficiency, invariably affect passengers, necessitating a robust system to manage and mitigate the impact.

This report focuses on developing and implementing an **advanced and robust hybrid Classical-Quantum Solution** for efficient re-accommodation of impacted passengers due to flight cancellations.

3 Background Literature review

The problem is effectively the famous **Passenger Recovery Problem (PRP)** [1], which addresses the recovery of passenger itineraries in the light of canceled/delayed flights. The problem has been studied as a classical optimization problem before, formulating the problem as a **mixed-integer programming model** [2].

On the quantum side, we analyzed the approaches used to solve various optimization problems inspired by the **Mphasis Crew Rostering** [3].

Running airline operations smoothly and cost-effectively on such a massive scale requires intricate planning and scheduling. Therefore, airline companies often use optimization tools to achieve this end in passenger re-accommodation. Specifically, the passenger recovery problem is a computationally intensive problem with large datasets and numerous complex constraints.

4 Problem Statement

4.1 Problem Formulation

The input-output structure is as given as follows for this problem statement

1. **Input:**
   (a) List of all PNRs (passenger name records)
   (b) List of all flights and their schedules
   (c) List of disrupted flights
   (d) List of business rules
   (e) List of penalty scores
2. **Output:**
   (a) List of re-accommodated PNRs along with updated itineraries
   (b) List of un-accommodated PNRs
   (c) Email notifications to impacted PNRs regarding changes in itinerary or cancellation

To manage input and output, a **business rule engine** is required. This engine allows the user (typically an airline) to input data, flexibly adjust business rules and penalty scores, and receive the corresponding output.

4.2 List of Flight Disruptions

1. **Schedule Duration Change:** This is addressed by treating the situation as a cancellation of the original flight, followed by the addition of a new flight at the adjusted times.

2. **Schedule Deletion (Cancelled Flight):** This scenario is straightforwardly handled as a case of a cancelled flight.

3. **Aircraft Type Change:** In this instance, the process involves cancelling the initial aircraft allocation and subsequently adding a new aircraft of the changed type to the schedule at the same time.

4. **Frequency of Schedule Change:** This is managed by cancelling the flight in the older schedule and then adding a new flight with the altered timing.

Thus, all the types of disruptions can be **reduced** to **flight cancellations**. Hence in our approach, we cancel **one** or **many** flights in the dataset and propose a solution.

4.3 Requirements

Each alternate flight solution/ reassignment has the following additional requirements .

1. The re-accommodation should be done at PNR level, i.e. all PAX (number of passengers) of an impacted PNR should be provided accommodation in same physical space, i.e. Same Aircraft and Cabins.

2. We restrict the number of connecting flights proposed in a solution to avoid offering absurd solutions to passengers.

3. The solution having the same departure and arrival city, is a soft constraint to ensure maximum convenience for the passengers however solutions with flights to nearby airports are preferred over not assigning a passenger.

4.4 Business Challenges

Handling the complexity of this problem requires addressing a wide range of operational and rule-based challenges, which can be summarized as follows:

1. **Multi-City Bookings:** This edge case arises when a PNR includes multiple bookings spanning a time greater than MAXCT (Maximum Connection Time) window.
   E.g., PNR P has a booking from Airport A to Airport B on Day X and another from Airport B to Airport C on Day Y, which is after Day X.
   The challenge here is to generate assignments that adhere to the ETD (Estimated Time of Departure) constraints as mentioned in the rule set. Additionally, if only one segment of the booking is disrupted, the newly proposed schedule should not negatively impact the other bookings within the same PNR.

2. **Round Trip Bookings:** This situation involves a PNR with bookings from an origin airport to a destination and then back to the original airport.
   E.g., PNR P has a booking from Airport A to Airport B and then from Airport B back to Airport A.
   The associated challenge is ensuring that the algorithm can efficiently handle these self-looping scenarios (where the initial source and final destination are the same) without introducing any errors or inefficiencies.

3. **Connecting Flights:** This edge case pertains to a PNR that includes multiple flight legs within the MAXCT time window.
   E.g., PNR P has flights from Airport A to Airport B, and from Airport B to Airport C, with the layover time between the airports falling between MCT (Minimum Connection Time) and MAXCT.
   The core challenge is to develop an algorithm capable of proposing alternative flights that comply with both the MCT and MAXCT constraints.

4. **1-1, Multi-1 Assignment Preference:** The problem statement specifies that 1-1 and Multi-1 assignments should be preferred in cases involving connecting flights.
   Therefore, the algorithm should be designed to prioritize these types of assignments, when feasible, over Multi-Multi and 1-Multi assignments. This ensures optimized scheduling in line with the given criteria.

5 Solution Methodology

Our approach uses a hybrid classical-quantum approach to solve the passenger re-accommodation problem. Our interface **provides flexibility** for the airline to make changes to their default rule set for flight ranking, passenger ranking, cabin upgrading and downgrading, etc. which plays a vital role in the objective function of our model. We also provide a **mailing option** for the airline to gather passenger preferences and finally rank different solutions and export them in a file. All the steps have been summarized in Figure 1. The steps of our algorithm are described in detail in the following subsections:

5.1 Identification of Impacted PNRs

We first identify all the impacted PNRs for whom any flight in their itinerary is disrupted. At this stage, we handle the Round-Trip Bookings and Multi-City Bookings by splitting the PNRs if the time between two subsequent flights in their itinerary is greater than MAXCT window since it is the maximum time till which a flight can be considered as a connecting flight.
E.g. Consider a scenario where a `PNR P` includes a round-trip booking from `MAA` to `BOM`, departing on `19-12-2023 22:30` Hrs, and a return flight from `BOM` to `MAA` departing on `20-12-2023 20:45 Hrs`. In our approach, this `PNR P` is divided into two separate entities:

- `PNR P#0`, encompassing the outbound flight from `MAA` to `BOM` scheduled for `19-12-2023 22:30 Hrs`.
- `PNR P#1`, comprising the return flight from `BOM` to `MAA` on `20-12-2023 20:45 Hrs`.

Each of these split PNRs is then processed individually within our algorithm. This splitting methodology is similarly applied to multi-city bookings. This approach not only aids in algorithmic processing but also provides valuable insights to the airline regarding the volume and nature of multi-city and round-trip bookings.

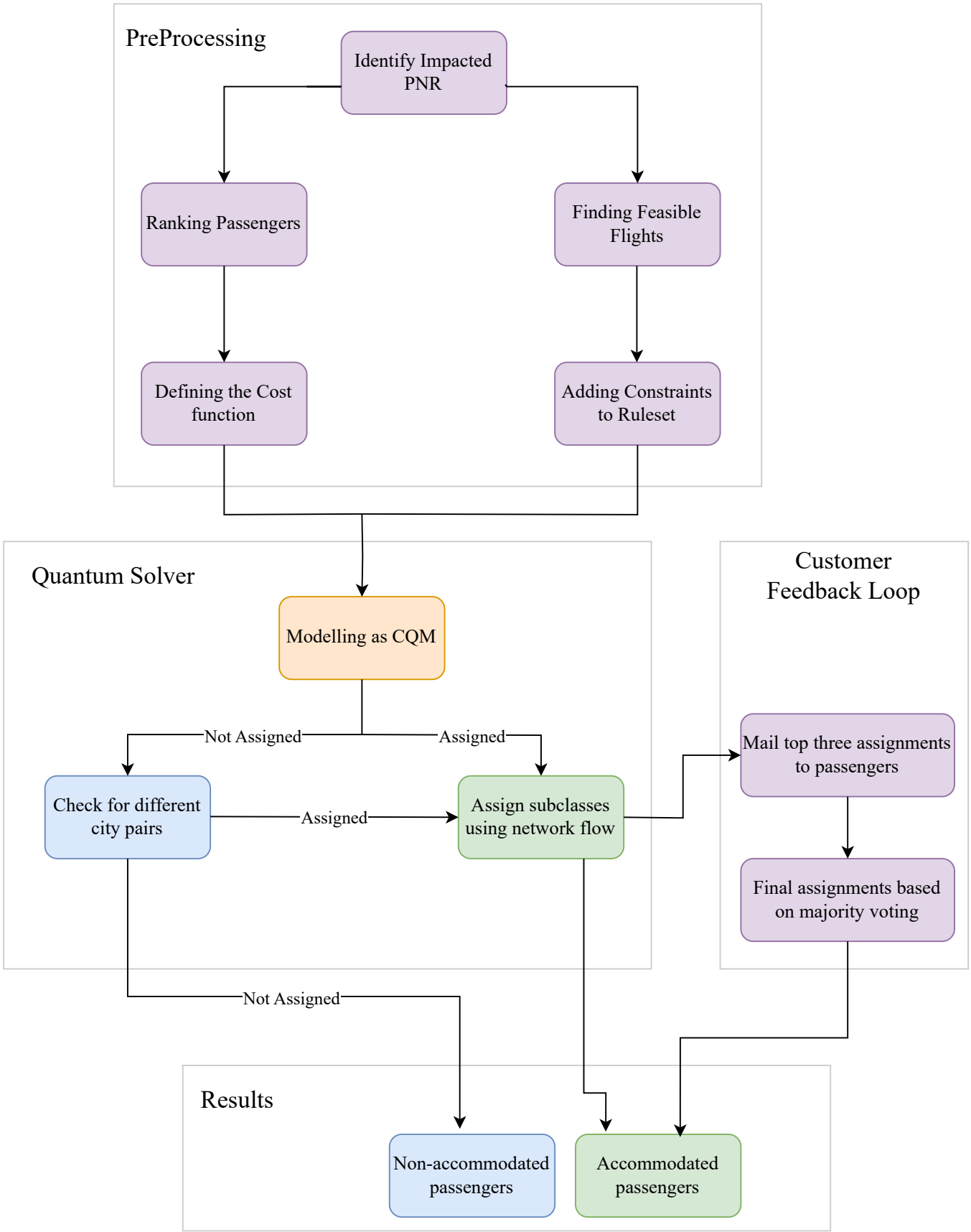


Figure 1: Workflow of the proposed solution

5.2 Development of the Airline Route Graph

Employing the NetworkX Python library, we constructed an aviation route graph, modeled as a Multi DiGraph(as there can be multiple flights between two airports). Airports are represented as nodes, identified by their `IATA` codes. For each air route connecting airport `A` to airport `B`, a directed edge from `A` to `B` is inserted into the graph as shown the example graph generated in Figure 2.

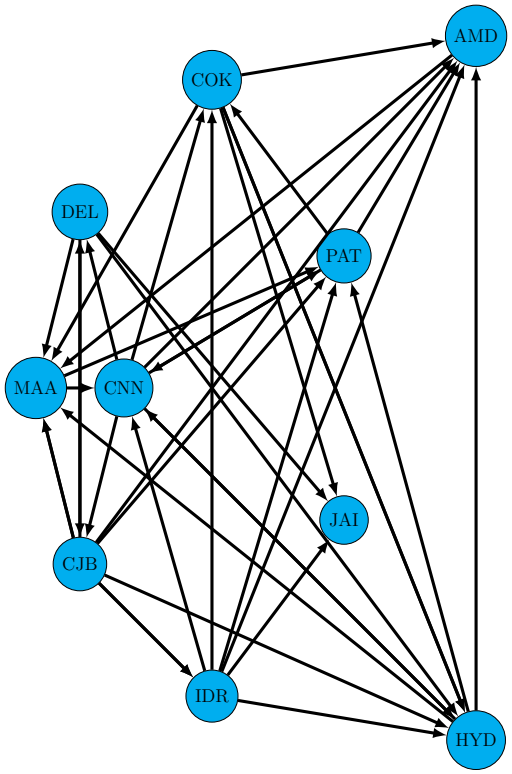


Figure 2: Flight Di-Graph

5.3 Identification of Alternate Feasible Flights

Following the graph construction, each entry in the Impacted PNR list is processed to identify alternate flight routes, employing a modified Depth First Search algorithm on the graph. Flight paths are computed from the departure airport of the **earliest** disrupted flight from the list of scheduled flights for this PNR to the final destination with the number of connecting flights limited to a maximum of $k = 4$ (configurable by the airline).

This method addresses the Generation of 1-1, Multi-1 options, as it searches for all feasible paths within the set length constraint, **adjustable** by the airline. Our algorithm's **efficiency** is notably enhanced through:

- **Dynamic Programming:** To circumvent redundant computations for PNRs with identical disrupted departure and arrival airports, we integrate **memoization** and **backtracking**. This strategy leads to a significant improvement in **time complexity**.
- **Prune and Search:** We visit an edge to be included in path only if it satisfies MCT,MAXCT, ETD conditions, otherwise it is not visited. This reduces the original time complexity of algorithm from $O(n!)$ to $O(\text{len(Feasible Paths)})$.

- **Iterative Methodology:** We replace the recursive approach of the Depth First Search algorithm with an **iterative** method using stacks, enhancing **scalability** for large datasets of flights.

Upon identifying potential routes, we once again evaluate their feasibility by **verifying adherence** to Minimum and Maximum Connection Time (MCT and MAXCT) criteria among successive flights, and by aligning with the ETD guidelines for the entire trip, thus conforming to the established rule-set. In instances where the PNR undergoes **segmentation** as described in the example in Identification of Alternate Feasible Flights, we ensure that the proposed alternative flight not only complies with MCT, MAXCT, and ETD requirements in isolation but also maintain these standards in relation to subsequent flights within the split itinerary.

5.4 Cost Function

We have chosen the cost function for a PNR to itinerary mapping as:

$$a * \log(s_1) + b * \log(s_2) + c * \log(s_3) \tag{1}$$

where,

1. $\mathbf{s_1}$ is the **flight quality score** of the proposed itinerary, which depends on **three parameters**:
   (a) **Arrival delay** at the final destination
   (b) **Departure delay** from the start airport, **as per industry standards**, we have **added more penalty for preponing the schedule** for the PNR in comparison to postponing the schedule. It also makes sense practically, as it is difficult to prepone schedules compared to postponing them.
   (c) **Connection Score**: which depends on **Number of connections** in the journey, so as to **reward less number of connections**, the score is calculated as:

$$\text{Connection Score} = \texttt{CC} - \texttt{PL} + \texttt{OL} \tag{2}$$

   where:

   `CC` : Connection Constant(CC)
   `PL` : Number of flight hops in proposed itinerary
   `OL` : Number of flight hops in original itinerary

   This ensures that the **score increases** if the length proposed itinerary is **as small as possible**, rewarding 1-1 and Multi-1 solutions. `Connection Constant` is a constant to normalize the score.

   We define the Delay Score as:

$$\text{Delay Score} = \text{Arrival Delay} + \text{Departure Delay} \tag{3}$$

   We have taken s_1 as:

$$s_1 = \sigma(\text{Delay Score} * \text{Connection Score}) \tag{4}$$

   where the σ is the sigmoid of function. The product of the two quantities is taken so that one quantity doesn't overshadow the other.

2. $\mathbf{s_2}$ is the **PNR Ranking Score** of the given PNR, which depends on **three parameters**:
   (a) **Special services request** (SSR) score, which has been divided into two categories, one with **medical requests**, one with **food requests**, for the airlines to define the score of SSR based on the category
   (b) **Loyalty score** based on the classes `CM, Platinum, Gold, Silver`
   (c) **PAX Score**, which is the score based on the number of passengers booked on the given PNR

   All three scores have been normalized using the min-max normalization.
   We have taken s_2 as:

$$s_2 = \sigma(\text{SSR Score} + \text{Loyalty Score} + \text{PAX Score}) \tag{5}$$

   where the σ is the sigmoid of function.

3. $\mathbf{s_3}$ is the **Class Quality Score**, which is calculated based on whether the PNR is getting a net upgrade or downgrade in the proposed itinerary. This is found out by calculating the average cabin score of the passenger in the proposed and original itinerary. The average cabin score is nothing but the weighted average of the cabins occupied by the PNR in the flights of an itinerary. The weights of each cabin (`FC, BC, PC, EC`) is based on the industry standard [4].
   - As roughly the price of the ticket of `EC` is the cheapest, it is assigned a weight of `1`.
   - As roughly the price of the ticket of `PC` is `1.5` times the price of `EC` ticket, it is assigned a weight of `1.5`.
   - As roughly the price of the ticket of `BC` is `2` times the price of `PC` ticket, it is assigned a weight of `3`.
   - As roughly the price of the ticket of `FC` is `2` times the price of `BC` ticket, it is assigned a weight of `6`.

   We have penalized downgrades more than upgrades, keeping in mind the concept of **customer retention**, because the booking of the Cabins is based on the luxury of that particular Class and a person is always prepared for an upgrade and never prepared for a downgrade. Again, s_3 is based on the sigmoid function.

4. **a**, **b**, and **c** are the weights of all the scores, and the weights are configurable by the user, improving the flexibility of the model. The sum of the weights can also be fixed so that we can compare the relative magnitude of the weights.

Why use logarithm and sigmoid functions?

We have used the sigmoid function, to normalize all the scores to a common level, so only the change of weights a, b, and c changes the importance of any score.

The use of logarithm in the cost function is quite strategic as its rate of change is not constant. This helps when comparing the change in the situation of two PNR itineraries. Any change in the situation of a low PNR-itinerary score should be penalized/rewarded more compared to the change in the situation of a high PNR-itinerary situation, as things become increasingly critical as the PNR-itinerary score decreases, for example:

- If the delay of a PNR with a high delay is reduced, and the delay of a PNR with a low delay is also reduced, the change in cost function can't be the same for both the PNRs, and the change should be more in the former case.

the logarithm fits all the requirements for the said usage, as it is a concave function.

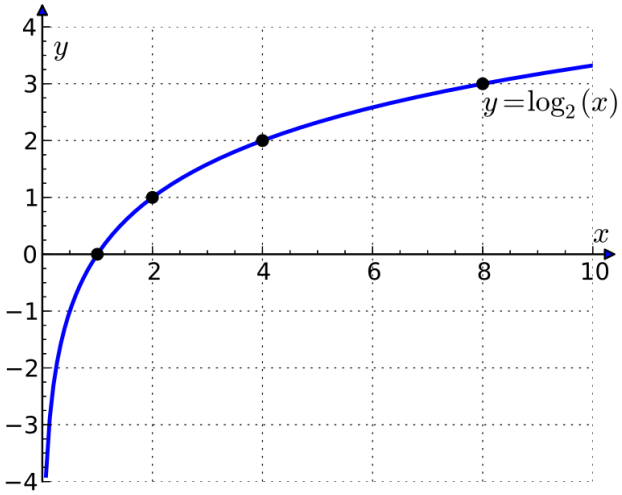


Figure 3: Logarithm is a concave function

5.5 CQM Modelling of the Problem

Constrained Quadratic model(CQM) can be defined mathematically as:

Minimize an objective:

$$\sum_i a_i x_i + \sum_{i \le j} b_{ij} x_i x_j + c,$$

Subject to constraints:

$$\sum_i a_i^{(m)} x_i + \sum_{i \le j} b_{ij}^{(m)} x_i x_j + c^{(m)} \circ 0, \quad m = 1, \ldots, M,$$

$\{x_i\}_{i=1,\ldots,N}$ can be binary, integer, or continuous variables, a_i, b_{ij}, c are real values, $\circ \in \{\geq, \leq, =\}$ and M is the total number of constraints.

The `ConstrainedQuadraticModel` class offered by `dimod` can contain this model and its methods provide convenient utilities for working with representations of a problem.

5.5.1 Objective Function

$$\text{Maximize:} \quad \sum_{i \in I} \sum_{j \in F_i} \sum_{k \in K} C_{ijk} X_{ijk} + \sum_{i \in I} (\text{Non-Assignment-Cost} \times (1 - \sum_{j \in F_i} \sum_{k \in K} X_{ijk}))$$

- $\mathbf{C_{ijk}}$ represents the cost to accommodate the i^{th} PNR to the k^{th} cabin of the j^{th} flight.
- $\mathbf{X_{ijk}}$ is a decision variable that is 1 when the i^{th} PNR is assigned to the k^{th} cabin of the j^{th} flight.

- **I** represents the set of all Impacted PNRs.
- $\mathbf{n_i}$ represents the number of passengers in the i^{th} PNR.
- **K** represents the set of all cabins.
- $\mathbf{b_{jk}}$ represents the available inventory in the k^{th} cabin of the j^{th} flight.
- $\mathbf{F_i}$ represents the set of Feasible Flights for the i^{th} PNR.
- **Non-Assignment-Cost** is a configurable constant and represents the penalty applied when a PNR is not accommodated successfully.

The first part of the objective function Maximize $(\sum_{i\in I}\sum_{j\in F_i}\sum_{k\in K} C_{ijk}X_{ijk})$ signifies the reward associated with assigning a passenger i.e the assignment cost.

The second part of the objective function $\sum_{i\in I}$ Non-Assignment-Cost $*(1-\sum_{j\in F_i}\sum_{k\in K} X_{ijk}))$ denotes the penalty associated whenever the model is unable to assign a PNR as the Non-Assignment-Cost will be negative.

Since the standard CQM has a minimization objective we simply multiplied the objective function with -1 to convert this problem into a minimization problem.

$$\text{Minimize:} \quad (-\sum_{i\in I}\sum_{j\in F_i}\sum_{k\in K} C_{ijk}X_{ijk} - \sum_{i\in I} \text{Non-Assignment-Cost} * (1 - \sum_{j\in F_i}\sum_{k\in K} X_{ijk}))$$

5.5.2 Constraints

Using Constrained Quadratic Modelling (CQM), we try to map every Impacted PNR to one of its feasible alternate flights till the Cabins level. The problem has two **hard constraint**, constraints that are not to be violated in any solution.

Capacity Constraint

This constraint ensures that no flight is booked beyond its available inventory:

$$\boxed{\sum_{i\in P_j} X_{ijk} * n_i \leq b_{jk} \;\; \forall k \in \text{K} \quad \forall j \in \text{F}}$$

- $\mathbf{P_j}$ represents the set of PNRs for which Flight j is feasible.
- **F** represents the set of all Flights that are feasible for atleast one PNR.

For modeling the constraint on a standard CQM it was formulated as:-

$$\sum_{i\in P_j} X_{ijk} * n_i - b_{jk} \leq 0 \;\; \forall k \in \text{K} \;\; \forall j \in \text{F}$$

Assignment constraints

This constraint ensures that no PNR is assigned the same flight twice:

$$\boxed{\sum_{j\in F_i}\sum_{k\in K} X_{ijk} \leq 1 \quad \forall i \in \text{I}}$$

For modeling the constraint on a standard CQM it was formulated as:-

$$\sum_{j\in\ F_i}\sum_{k\in K} X_{ijk} - 1 \leq 0 \quad \forall i \in \text{I}$$

5.6 Quantum Strategy using D-Wave

1. To solve the problem using Quantum Computing, the problem was Modelled using the powerful [5] **CQM**(Constrained Quadratic Model) by **D-Wave** instead of the old **BQM** (Binary Quadratic Model) by D-wave due to the following reasons:
   (a) BQM doesn't support making constraints inherently. We worked around this fact by introducing **penalty scores** for breaching some conditions, even then BQM couldn't solve our problem for large datasets.
   (b) BQM Solver frequently violated the feasibility constraints.
   (c) Moreover the running time of BQM Solver was way worse than the CQM Solver as shown in Table 7.
2. The model was **pre-processed** using D-wave's presolver to reduce the number of variables before passing the CQM to the solver thereby reducing the complexity of the model and a reduced run time on the hybrid solver.
3. LeapHybridCQMSampler was used to sample the solutions of the problem and generate all the **low energy solutions**, the sample set was then filtered based on the problem constraints to obtain all the **feasible low energy solutions**.

The LeapHybridCQMSampler works on this methodology. The model extracts the top 3 low-energy solutions from the set of feasible solutions returned by the LeapHybridCQMSampler. At this stage of the pipeline, majority of the impacted PNRs have been successfully mapped to an alternate **Flight and a Cabin** obeying all constraints.

5.7 Consideration of different City Pairs

In cases where the algorithm leaves some PNRs without suitable accommodations, alternative resolutions are sought by considering varying city pairs. For each disrupted PNR, we identify the $k = 3$ closest neighbouring airports to its intended destination using KDTree data-structure and utilizing **real-time traffic data** and the **travel time** between the original and proposed airports, as determined by the Distance Matrix API Accurate. We **exclude** any alternative city pair if the travel duration from the original to the proposed airport exceeds the CP-THRESH limit. When multiple city pair alternatives are available for an impacted PNR, they are ranked and scored according to the following equation:

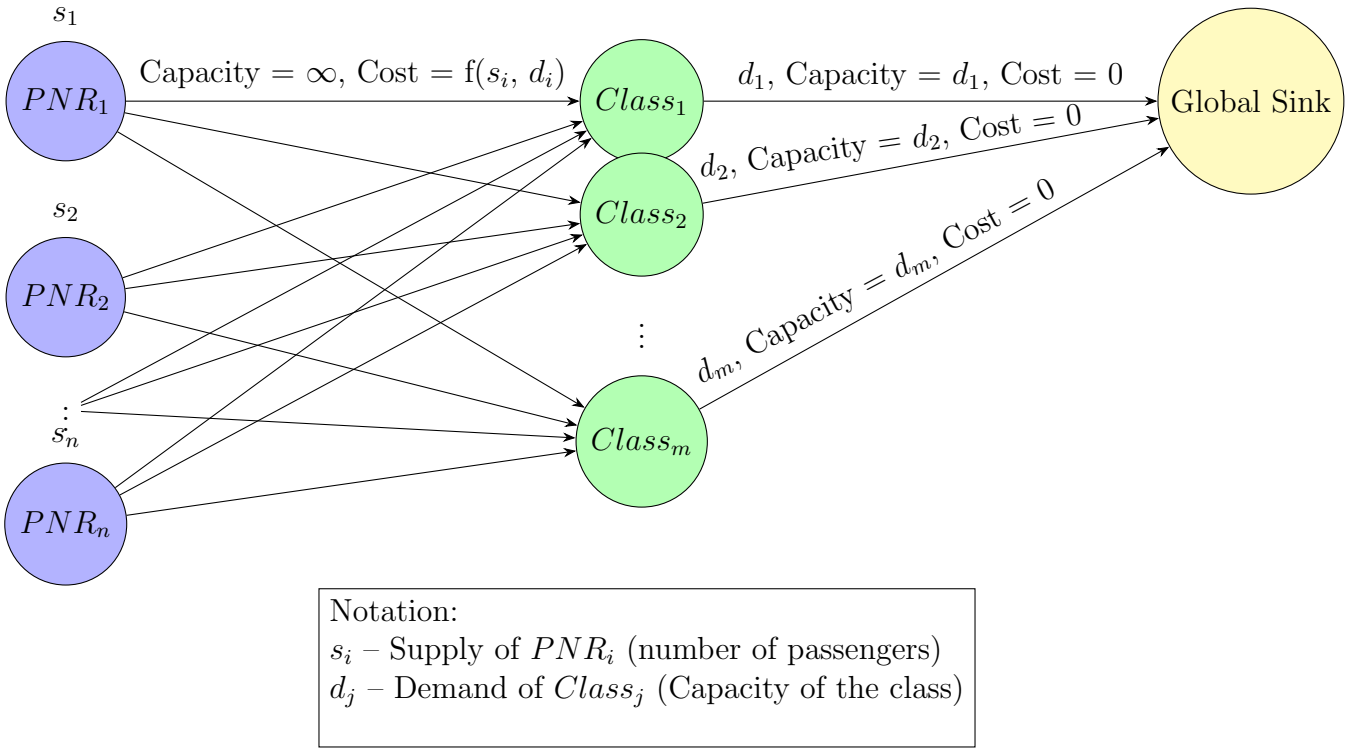
$$\frac{1}{d^{\alpha} + t^{\beta} + \epsilon} \tag{6}$$

where:

d : Distance of the road journey between the initial and the proposed airport
t : Time taken in real-time to travel from the initial airport to the proposed one
ϵ : A negligible value to prevent overflow errors
α, β : Parameters to balance the significance of d and t in the scoring formulae

This scoring mechanism favours alternatives that are **closer** and require **less travel time**. Subsequently, the methodology outlined in the preceding sections is replicated, adapting the arrival airport to the new proposed airport(s).

Figure 4: Network Flow Graph



5.8 Quantum Network Flow

After we have assigned a Cabin and a Flight to each of the PNRs (done at the PNR level), we need to assign Class to **each of the passengers** in the PNR inside that particular cabin. This is done with the use of Network Flow, specifically **min-cost-max-flow**.

1. **Flow Network Structure:** The system can be elegantly represented as a **fully connected bi-partite graph**. In this model, the nodes on the left symbolize the PNRs assigned to a specific Cabins, while the nodes on the right correspond to the Classes within that Cabin. Each PNR node is assigned a 'supply' value, equivalent to the number of passengers in the PNR. Conversely, the Class nodes reflect the 'max-capacity,' representing the available seats remaining in each class.

   Illustrated in Figure 4, the structure of the Flow-graph demonstrates this setup.

   Each Class node connects to a global sink via edges with capacities equal to the remaining capacity of the respective class. These edges, assigned a zero cost, ensure that all flow reaching a class must ultimately go to the global sink. This design places an upper bound on the max-flow that can reach each class node as the capacity of that edge is the remaining capacity of that class node. For instance, if there are two classes with remaining capacities of 3 and 5, two edges are constructed with capacities 3 and 5, respectively, and a cost of 0 each, connecting each class to the sink.

2. **Cost Functions:** Each of the inner edges of the fully connected bi-partite graph has a cost which is the loss that will occur or the negative of the score that the airline gets if a particular passenger of that PNR is assigned to the corresponding Class. Our **min-cost max flow** algorithm (Classical approach) will calculate the **maximum flow (assigns every PNR a Class) with minimum cost or the maximum score**. First, we solved it classically then we formulated it as a **CQM** [6] problem and solved it using **quantum flow.**

- Our business rule engine assumes a Class-Class mapping which has to be followed during the change of classes as a **soft constraint**.
- For defining our cost function, we do the following:
  Let C_{orig} be the original class of a passenger, and C_{new} be the proposed new class. The airline provides a mapping M that specifies acceptable class changes, where each class C is mapped to a set of classes $M(C)$ to which a passenger can be reassigned without significant penalty.

  The cost function f for reassigning a passenger from C_{orig} to C_{new} is defined as follows:

  – If $C_{\text{new}} \in M(C_{\text{orig}})$, the new class is in the approved mapping of the original class. In this case, the cost is decreased to reflect a favorable class change. The cost function is given by:

  $$f(C_{\text{orig}}, C_{\text{new}}) = -\texttt{PNR Score} \times \texttt{Class Change Constant}$$

  – If $C_{\text{new}} \notin M(C_{\text{orig}})$, the new class is not in the approved mapping of the original class. In this case, the cost is increased to penalize the unfavorable class change. The cost function is defined as:

  $$f(C_{\text{orig}}, C_{\text{new}}) = +\texttt{PNR Score} \times \texttt{Class Change Constant}$$

  Here, `Class Change Constant` is a predefined constant that quantifies the cost of changing the class.

3. **Quantum Network Flow using CQM:** We can convert this into a **CQM** and solve using **CQM** solver using quantum computing.

$$\begin{aligned}
&\text{Objective Function:} \quad \text{Minimize} \left(\sum C_{ij} X_{ij} \right) \\
&\text{Supply Constraint:} \quad \forall i \in \text{PNRs} \quad \sum_j X_{ij} = n_i \\
&\text{Demand Constraint:} \quad \forall j \in \text{Classes} \quad \sum_i X_{ij} \leq b_j
\end{aligned}$$

where,

- $\mathbf{C_{ij}}$ represents the cost(loss) to accommodate the i^{th} PNR's passenger to the j^{th} Class.
- $\mathbf{X_{ij}}$ represents number of passengers of the i^{th} PNR who were accomodated in j^{th} Class.
- $\mathbf{n_i}$ represents the number of passengers in the i^{th} PNR.
- $\mathbf{b_j}$ represents the number of remaining seats in the j^{th} Class.

Objective function:- As it is a min-cut max flow problem, we are trying to minimize the costs or the negative of the scores. Since $\mathbf{C_{ij}}$ is the cost per unit passenger if i^{th} PNR's passenger to the j^{th} Class, we multiply it by number of passengers going to that Class and add it to the total cost.

The **Supply Constraints** ensures that every passenger of a PNR is mapped to a particular Class or mathematically **incoming flow = outgoing flow**(conservation of flow).

The **Demand Constraints** ensures that the maximum capacity of a class is not violated while accommodating the passengers.

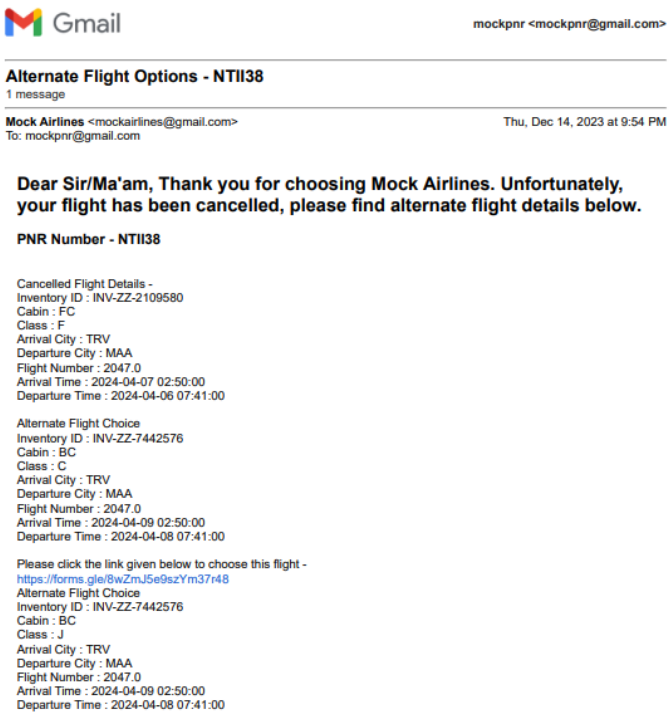
Thus by solving the above formulation in **CQM** solver of **DWave**, we can find the optimal value of the variables $\mathbf{X_{ij}}$'s and from that retrieving the optimal mapping is trivial, for example: Assume that $X_{12} = 5$, this implies that 2 passengers of PNR = 1 are assigned to Class numbered 2 and so on similarly.

4. **Concurrent Processing:** As, the task of assigning Classes to PNRs of each pair of flight-Cabin tuple is **independent**, we utilize **parallel processing** to speedup the operation. Data integrity is safeguarded through using **Locks** in shared data structures.

5.9 Customer Feedback

The quantum solver presents three leading solutions, each associated with specific class assignments via the **Quantum Network Flow** algorithm. The **Majority Voting Algorithm** [7] is integral here, as it integrates **customer feedback** into the decision-making process.

Each accommodated PNR is sent an **email** (Refer Figure 5) with these three flight options. Passengers' preferences are meticulously evaluated, with each solution gaining a weighted score based on their PNR rankings. The solution that gets the most votes, reflecting the majority preference, is then selected as the final decision.


Gmail mockpnr <mockpnr@gmail.com>

Alternate Flight Options - NTII38
1 message

Mock Airlines <mockairlines@gmail.com> Thu, Dec 14, 2023 at 9:54 PM
To: mockpnr@gmail.com

Dear Sir/Ma'am, Thank you for choosing Mock Airlines. Unfortunately, your flight has been cancelled, please find alternate flight details below.

PNR Number - NTII38

Cancelled Flight Details -
Inventory ID : INV-ZZ-2109580
Cabin : FC
Class : F
Arrival City : TRV
Departure City : MAA
Flight Number : 2047.0
Arrival Time : 2024-04-07 02:50:00
Departure Time : 2024-04-06 07:41:00

Alternate Flight Choice
Inventory ID : INV-ZZ-7442576
Cabin : BC
Class : C
Arrival City : TRV
Departure City : MAA
Flight Number : 2047.0
Arrival Time : 2024-04-09 02:50:00
Departure Time : 2024-04-08 07:41:00

Please click the link given below to choose this flight -
https://forms.gle/8wZmJ5e9szYm37r48
Alternate Flight Choice
Inventory ID : INV-ZZ-7442576
Cabin : BC
Class : J
Arrival City : TRV
Departure City : MAA
Flight Number : 2047.0
Arrival Time : 2024-04-09 02:50:00
Departure Time : 2024-04-08 07:41:00

Figure 5: Our business rule engine mails the impacted PNRs their proposed schedule and asks for their choice.

6 Flexibility

Our solution stands out for its flexibility, particularly in **accommodating various business rules**, and allowing airlines to **modify** scores and parameters to obtain tailored solutions.
For example, airlines are provided with options to allow for the **upgrade or downgrade of**

cabins through our business engine adding another layer of customization to the solution. For more, refer Figure 10, 12, and 11 of Appendix C.

Furthermore, our system offers flexibility in adjusting the **CC**. For instance, setting a lower CC tends to prioritize flights with fewer hops, even if they have delays. Conversely, increasing the CC shifts the preference towards flights with minimal delays, regardless of the number of hops involved. Consider the following impacted PNR given in Table 1 and the feasible flights, including the cancelled ones, are listed in Table 2:

Table 1: PNR

PNR_Number	Inventory_ID
PNR001	INV-ZZ-1875559

Table 2: Flights

Inventory ID	Departure city	Arrival city	Departure time	Arrival time	Status
INV-ZZ-1875559	MAA	CNN	2024-06-1 04:37:00	2024-06-1 6:50:00	Cancelled
INV-ZZ-1875562	MAA	CNN	2024-06-1 20:37:00	2024-06-1 22:50:00	On Time
INV-ZZ-1875563	MAA	GAU	2024-06-1 6:37:00	2024-06-1 9:50:00	On Time
INV-ZZ-1875564	GAU	CNN	2024-06-1 11:37:00	2024-06-1 14:50:00	On Time

With a lower CC (between 0 and 1), the system favors direct flights, such as the one with Inventory ID `INV-ZZ-1875562`, despite its higher arrival delay. However, as the CC is increased beyond a certain threshold, the preference shifts to connecting flights, specifically those with Inventory IDs `INV-ZZ-1875563` and `INV-ZZ-1875564`.

Table 3: At higher CC, lower arrival delay is given more priority

PNR_Number	Inventory_ID
PNR001	INV-ZZ-1875563, INV-ZZ-1875564

Table 4: At low CC, lower number of flights is given more priority

PNR_Number	Inventory_ID
PNR001	INV-ZZ-1875562

Thus, airlines can fine-tune various scores, rules, and preferences based on different factors, demonstrating the solution's extensive adaptability and customizability. This adaptability makes our solution a robust tool in the dynamic airline industry.

7 Results

Results in table 5 have been generated with our **Hybrid Classical-Quantum Model** outlined in previous sections on two datasets, one containing **603 flights** and another containing **2012**

Table 5: Solution Statistics, where the last two rows are from the dataset given by Mphasis.

Total Flights	Impacted PNR	Re-accom. PNR	# of Up-grades	# of Down-grades	# of Same City Pair	# of Different City Pair	Mean Arrival Delay (in hr)
603	87	69	37	13	69	0	22.023
2012	490	394	154	143	394	0	25.544
2012	629	629	331	165	622	7	30.356
2012	1233	1118	474	380	1109	9	31.563

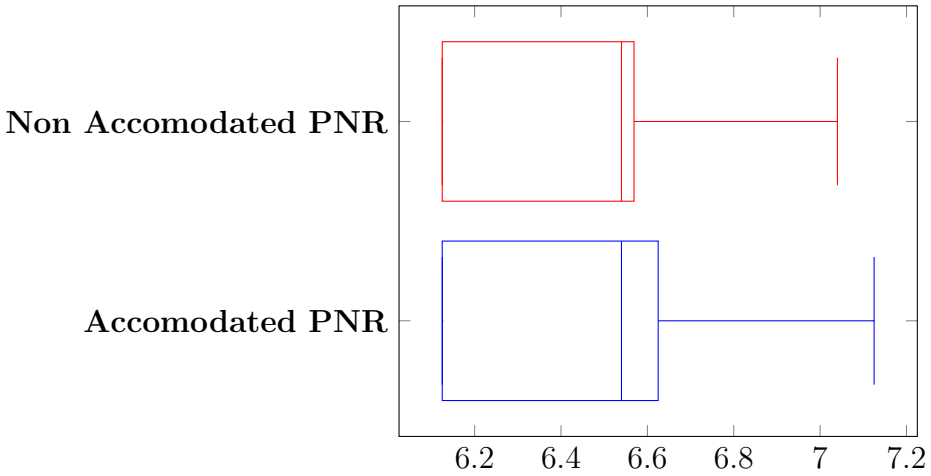


Figure 6: Box plot for distribution of PNR Scores

flights. We canceled some combination of flights and analyzed the results with upgrades, downgrades and different city-pairs **enabled**.

This demonstrates that our model exhibits **robust performance**, even when handling extensive datasets. It effectively optimizes outcomes by balancing the trade-offs among the number of re-accommodations, the average arrival delay, and the instances of passenger upgrades and downgrades.

The boxplot for them scores of impacted PNRs has been drawn as shown in Figure 6. We draw the following conclusions:

- Since the upper quartile [8] of the assigned PNRs (`6.625`)is higher than the upper quartile of non-assigned PNRs (`6.569`), it means more PNRs have a higher ranking in accommodated than non-accommodated (refer `row 4` of Table 5).
- For both distributions, the median [8] is the same (`6.54`), indicating that evenness is given.
- The upper whisker value [8] is higher for the accommodated PNR's box plot (`7.125` > `7.04`) which indicates a higher PNR being accommodated (`row 4` of Table 5).

8 Discussion and Scalability

We solved the problem as Multi Integer Programming (MIP) model classically using **Gurobi Optimization** solver. Since Gurobi is recognized for its exceptional performance as an MIP solver [9], we deliberately chose Gurobi as the benchmark against which to evaluate DWave's LeapHybridSolver.

Note: We use **Average QPU Access Time** [10] and refer it as **Average Quantum Time** measure against the time taken to solve the problem classically by Gurobi optimizer. The following Table 6 gives the time for various parts of our pipeline: We drew the following

Table 6: Solver Time Data, where the last two rows are from the dataset given by Mphasis.

Total Flights	Net Impacted PNR	# of Variables	# of Constraints	Avg. Classical Time (in ms)	Avg. Quantum Time (in ms)	% of Abs. Difference in Score	Pre-Process Time (in s)	Network Flow Time (in s)
603	87	1352	103	995	16.03	0.003	0.7	0.023
603	518	8312	495	129.33	16.04	0.0005	3.43	0.139
2012	490	4844	533	3439.67	21.37	0.008	5.64	0.073
2012	1233	32768	1603	3623	16.05	0.36	17.96	0.292

conclusions from the above comparison:

- There is very small difference between the score that is maximised by classical and quantum.
- Beyond a specified threshold of impacted PNRs (e.g., `518` dataset), Gurobi **stops** yielding results without a purchased license. In contrast, DWave's LeapHybridSolver consistently produces results under its free developer access.
- The Quantum approach using DWave's LeapHybridSolver achieves the optimal result **faster** than classical approach using Gurobi as reported by the above table.

9 Conclusion

In this report, we address the Problem Statement by implementing a **robust business rule engine**. The designed engine exhibits a high degree of flexibility, accommodating dynamic rule changes and adjustments to penalties, as may be necessitated by end-users, typically within the context of an airline operation.

Our solution provides users with **alternative flight options**, ranking them based on metrics such as **total impacted PNRs**, **mean arrival delay**, and the **count of solutions falling into the categories of** 1-1, 1-Multi, and Multi-1.

To formulate this complex problem, we leveraged a CQM and employed **D-Wave's Leap Hybrid Solver**. Various techniques, including **quantum network flow** for class assignment, **KDTree** for identifying neighboring airports, and **Depth First Search** for pruning and searching the solution space, were rigorously applied.

Remarkably, our findings indicate that quantum computers excel in solving this problem, **outperforming classical computers** in terms of speed and efficiency.

A D-Wave Quantum Computers

DWave's hybrid solvers mix classical heuristic solvers and quantum processing units (QPU). When a problem is given to the solver, firstly the heuristic solvers act on it and then they **query** the QPUs for fast and effective results.

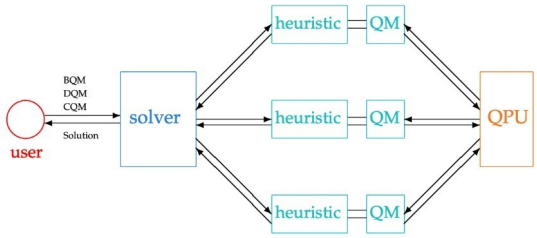


Figure 5: Structure of a hybrid solver in HSS. The front end (blue) reads an input Q and optionally a time limit T. It invokes some number of heuristic solvers (threads) that run on classical CPUs and GPUs (teal) and search for good-quality solutions to Q. Each heuristic solver contains a quantum module (QM) that formulates and sends quantum queries to a D-Wave QPU (orange); QPU responses to these queries may be used to guide the heuristic search or to improve the quality of a current pool of solutions. Before time limit T, the heuristic solvers send their results to the portfolio front end, which, for example, removes duplicates and forwards a subset of solutions to the user.

Figure 7: Working of DWave's hybrid solvers.

D-Wave provides a host of quantum computers that work on the principle of **quantum annealing** [11]. The architecture of these computers allows us to minimize the energy of an **Ising-like system**[12]. Each qubit is constructed using a **current-carrying superconducting loop**, where the magnetic field in the clockwise direction represents 0 and the anti-clockwise direction represents 1. Being a quantum object, the qubits exist in a **superposition of 0 and 1**. An external bias field and coupling between the qubits are controlled to model a particular quadratic problem in the hardware of the Quantum Processing Unit. The Hamiltonian can be represented as follows,

$$H = \underbrace{-\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\textit{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{ij}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\textit{Final Hamiltonian}}$$

where $\hat{\sigma}_{x,z}^{(i)}$ are the pauli matrices[13] acting on i^{th} qubit, J_{ij} is the coupling parameter and h_i is the external bias. The initial Hamiltonian has a well-known ground state where all possible qubit states have equal probability of occurrence. The final Hamiltonian is the answer to the problem that we are trying to solve. The annealing process takes place in the following fashion,

1. Initially, A(s) is set to a high value and B(s) is set to 0. In the ground state, all qubits are in the superposition state.

2. A(s) is slowly reduced and B(s) is increased. The effect of initial Hamiltonian is less pronounced and the system still remains in the ground state owing to the **adiabatic theorem**[14].

3. Running the annealing process too fast can lead the system to escape from the ground state. There could also be thermal fluctuations that disturb the system. For this reason, our problem gives multiple solutions that are not always perfectly optimal according to the Hamiltonian.

4. Finally, A(s) becomes 0 and all the qubits take one of the bit states 0 or 1. Since, it's a quadratic problem, there can only be one global optimum and thus, no superposition is possible in the ground state.

B BQM vs CQM

The below table shows the comparison of processing times of BQM and CQM for the problem. We use the **QPU access time** to compare both BQM and CQM solver of **DWave** because it is the time which is not affected by network and scheduling latencies (see Figure 8)

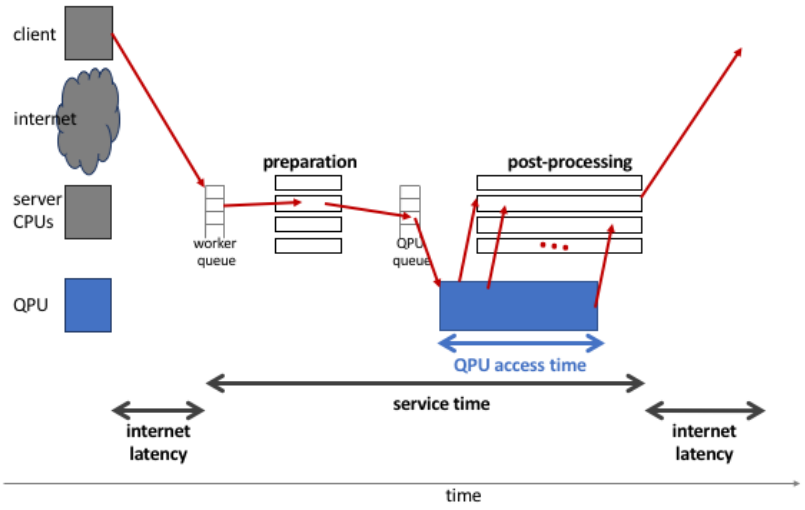


Figure 8: We follow the definition provided by **DWave** for measuring QPU time.

Table 7: Comparison of BQM and CQM, where the last row is from the dataset given by Mphasis.

Net Impacted PNR	BQM QPU time (in ms)	CQM QPU time (in ms)
87	42.747	16.03
490	128.151	21.37
518	170.875	16.04
1233	**Could not solve the problem**	16.05

C Pictures of the Business Rule Engine

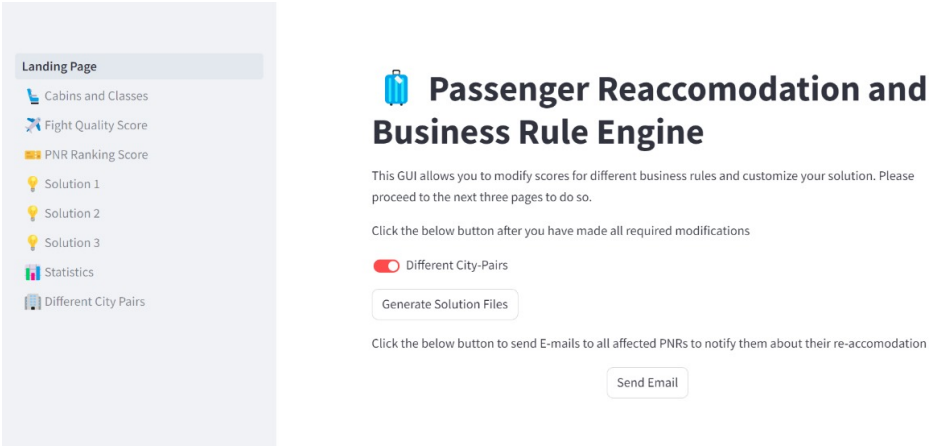


Figure 9: The landing page of our engine, you can generate the solutions, while opting for different city pairs or not, and also mail the impacted PNRs their changed itineraries.

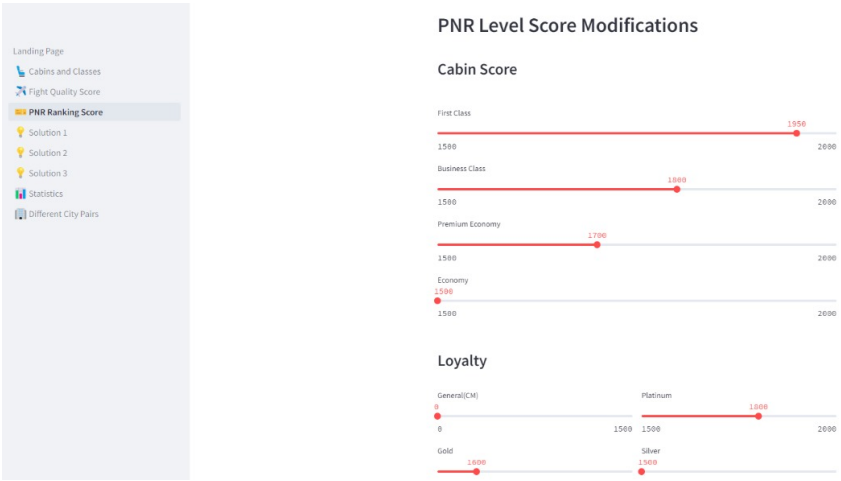


Figure 10: Our business rule engine allows for the modification of various scores related to PNR.

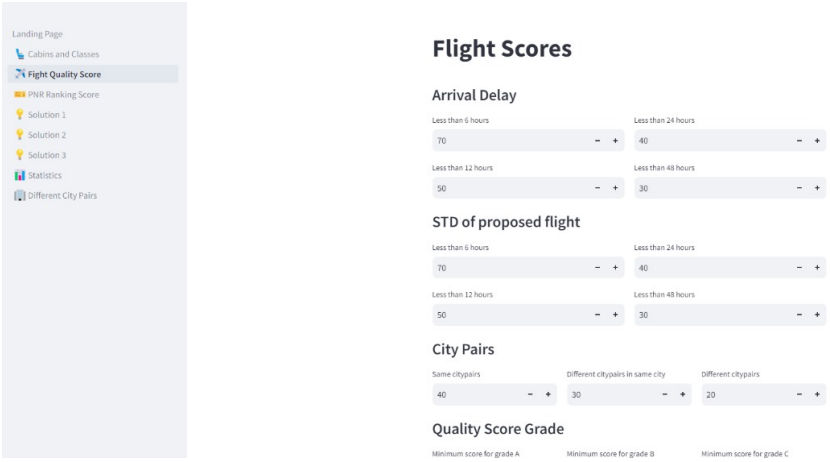


Figure 11: Our business rule engine allows for the modification of various scores related to PNR-itinerary mapping.

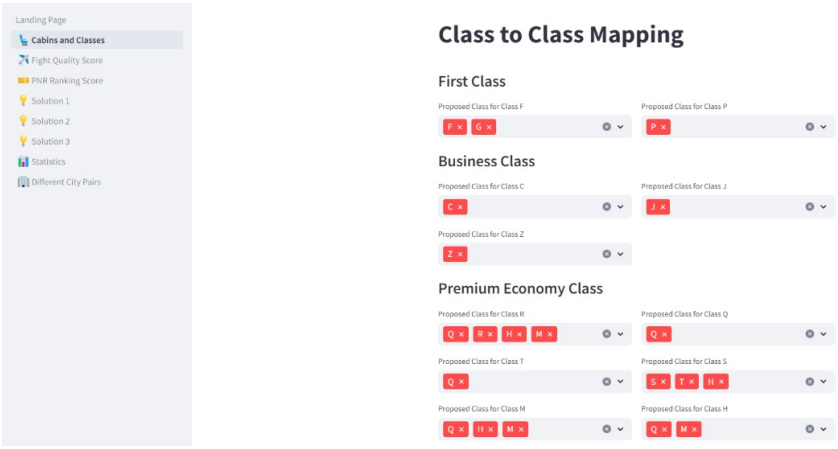


Figure 12: Our business rule engine allows for the modification of various scores related to Class-Class mapping.

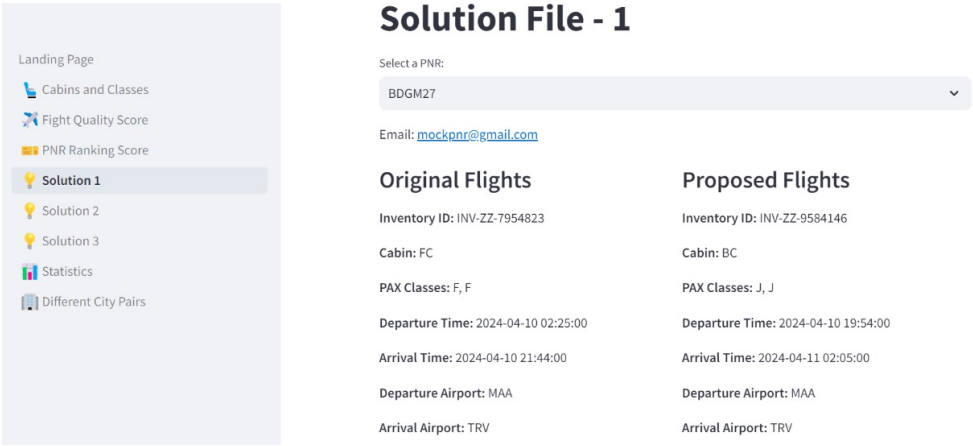


Figure 13: A Solution generated by our engine.

Figure 14: Our engine also shows the statistics related to the best solution generated.

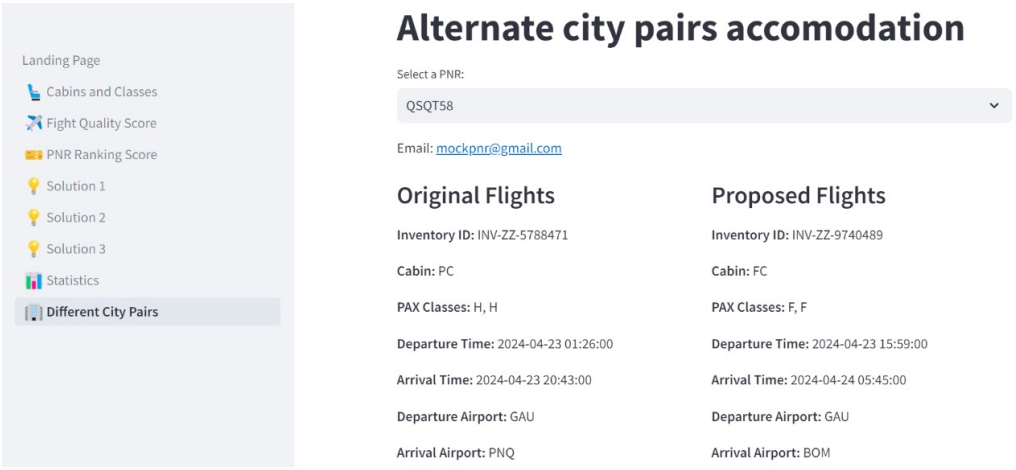


Figure 15: Our engine also shows the PNRs whose arrival city is different from what they had booked.

Glossary

1-1 Replacing every impacted flight with another **single-legged** flight. 4, 6, 7, 17

1-Multi Replacing every impacted flight with **multi-legged** flights. 4, 17

Cabin Cabins physically divide a plane into different categories, each with its own price and set of rules. There are four main cabins - First (FC), Business (BC), Premium Economy (PC) and Economy (EC). 3, 8, 10, 12, 14

CC "Connection Constant", a constant to normalize the connection score. 7, 15

Class Each cabin is logically further divided into various classes. They typically denote the level of service or fare type purchased by a passenger. The class often determines the amenities, seat comfort, services, and flexibility available during the flight. 8, 12–14, 21

CP-THRESH "City Pairs Threshold", indicates the maximum allowable travel time by road between the initial airport in the booking and the suggested airport in an alternative solution. 11

CQM D-Wave's CQM(Constrained Quadratic Model) supports problems of the form:

Minimize an objective:

$$\sum_i a_i x_i + \sum_{i \le j} b_{ij} x_i x_j + c,$$

Subject to constraints:

$$\sum_i a_i^{(m)} x_i + \sum_{i \le j} b_{ij}^{(m)} x_i x_j + c^{(m)} = 0, \quad m = 1, \ldots, M,$$

. 10, 12–14, 17

ETD Estimated time of departure of a planned flight. 3, 6, 7

hard constraint Constraints which can't be violated at any cost. 10

MAXCT Maximum Connection Time between two connecting flights. 3, 4, 6, 7

MCT Minimum Connection Time between two connecting flights. 4, 6, 7

Multi-1 Replacing a subset impacted flights with a **single-legged** flight. 4, 6, 7, 17

Multi-Multi Replacing a subset impacted flights with another **multi-legged** flights. 4

PAX PAX denotes the number of passengers attached to a single PNR. 3, 8

PNR Passenger Name Record data is unverified information provided by passengers and collected by air carriers to enable the reservation and check-in processes. The data is used by the air carriers to manage their air transportation services. 2–4, 6–14, 16, 17, 20, 22

soft constraint Constraints which may be violated, but there is a **penalty** for violating it. 3, 13

SSR SSR stands for Special Service Request. SSRs are codes used within the airline industry to communicate specific passenger or flight-related requests and information to the airline's reservation and operation systems. 8

References

[1] Stephane Bratu and Cynthia Barnhart. "Flight operations recovery: New approaches considering passenger recovery". In: *Journal of Scheduling* 9.3 (June 2006), pp. 279–298. ISSN: 1099-1425. DOI: 10.1007/s10951-006-6781-0. URL: http://dx.doi.org/10.1007/s10951-006-6781-0.

[2] Huanzheng Wang, Xingye Dong, and Youfang Lin. "A column-generation-based heuristic for passenger recovery problem considering multi-flights and seat-sharing constraints". In: *2020 International Conference on Control, Automation and Diagnosis (ICCAD)*. 2020, pp. 1–6. DOI: 10.1109/ICCAD49821.2020.9260523.

[3] Sidh Satam, Siddharth Shankar, and Rohit Patel. *Quantum Computing-based Airline Crew Rostering Optimization*. https://www.mphasis.com/content/dam/mphasis-com/global/en/home/innovation/next-lab/Airline_Crew_Rostering.pdf. [Accessed 15-12-2023].

[4] Marisa Garcia. *Calculating the Cost of a Better Airline Cabin for All — skift.com*. https://skift.com/2016/01/13/calculating-the-cost-of-a-better-airline-cabin-for-all/. [Accessed 12-12-2023].

[5] DWave Systems. *DWave's Leap Performance Update*. https://bit.ly/dwave-leap. [Accessed 16-12-2023].

[6] Thomas Krauss et al. "Solving the max-flow problem on a quantum annealing computer". In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–10.

[7] B. Parhami. "Voting algorithms". In: *IEEE Transactions on Reliability* 43.4 (1994), pp. 617–629. DOI: 10.1109/24.370218.

[8] *Understanding and interpreting box plots*. https://www.wellbeingatschool.org.nz/information-sheet/understanding-and-interpreting-box-plots. [Accessed 16-12-2023].

[9] Lars Magnus Hvattum, Arne Løkketangen, and Fred Glover. "Comparisons of commercial MIP solvers and an adaptive memory (tabu search) procedure for a class of 0–1 integer programming problems". In: *Algorithmic Operations Research* 7.1 (2012), pp. 13–20.

[10] DWave Systems. *Operation and Timing; D-Wave System Documentation documentation*. https://docs.dwavesys.com/docs/latest/c_qpu_timing.html. [Accessed 15-12-2023].

[11] DWave Systems. *What is Quantum Annealing? D-Wave System Documentation documentation — docs.dwavesys.com*. https://docs.dwavesys.com/docs/latest/c_gs_2.html.

[12] Johannes Obermeyer. *The Ising Model in One and Two Dimensions*. https://www.thphys.uni-heidelberg.de/~wolschin/statsem20_3s.pdf. 2020.

[13] Caltech Feynman Lectures. *The Pauli spin matrices*. https://feynmanlectures.caltech.edu/III_11.html#Ch11-S1.

[14] Tameem Albash and Daniel A. Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (Jan. 2018). ISSN: 1539-0756. DOI: 10.1103/revmodphys.90.015002. URL: http://dx.doi.org/10.1103/RevModPhys.90.015002.