Mphasis
The Next Applied

Quantum Computing-based Airline Crew Pairing Optimization

Whitepaper by NEXT Labs, the Mphasis research arm focused on disruptive and breakthrough innovations across Cognitive, Cloud and Service Transformation

Contents

1.
Executive Summary

The airline industry globally uses a suite of optimization tools for flight scheduling, fleet assignment, aircraft maintenance and crew scheduling. These tools enable reliable, efficient and time-critical operations and cost reduction. With the increased availability of quantum computing-based optimization solutions, airlines can upgrade these high-impact tools to deliver superior performance in both run time and solution quality.

In this paper, we use quantum optimization-driven designs for airline operations that can handle large-scale problems when compared against open-source classical and quantum-inspired optimization routines.

2.
Background

IATA (International Air Travel Association) and ICAO (International Civil Aviation Organization) are leading groups in the aviation and airline industry, which have assigned airline codes to more than 5,000 local, regional and international airlines. Before the COVID-19 crisis, 1,126 commercial airlines operated more than sixteen million flights, carrying 4.5 billion customers annually [1]. American Airlines, a major US (United States) airline, operates approximately 5900 flights per day (translating to ~70000 flights per month) with a ~900-strong fleet size [2]. Running airline operations smoothly and cost-effectively on such a massive scale requires intricate planning and scheduling. Airline companies often use optimization tools to achieve this end in flight scheduling, fleet assignment, aircraft maintenance and crew scheduling. Specifically, airline crew scheduling is a computationally intensive problem with large datasets and numerous complex constraints. It is typically solved sequentially as a two-part problem - crew pairing followed by crew rostering.

- **Crew pairing:** The system sequences the list of flight legs for a defined period and refers to each as a ‘crew pair’.
- **Crew rostering:** Crew rostering optimization uses flight pairings generated by flight pairing optimization, along with crew details such as base location and crew count, and assigns a pre-defined number of crew members to each pairing to create a ‘crew roster’.

In this paper, we apply a hybrid classical-quantum approach to solve the crew pairing problem. An airline crew pairing is a sequence of flight legs or duty periods within the same fleet, starting and ending at the same home crew base. This step comes after the fleet assignment in the airline resource planning process. Once crew pairings are created, airlines can assign crew members tasks on their designated flights.

The total crew operating costs (including salaries, benefits and expenses) are the second largest operating cost for an airline, after fuel costs [3]. Thus, even a small percentage of savings in crew

expenses through optimal crew scheduling translates to a sizable cost reduction for the airline [4]. This makes airline crew scheduling one of the most critical components of the airline scheduling process.

3.
Business Drivers and Challenges

Effective assignment of crews to a flight is an essential aspect of airline planning due to several factors:

- **Scale and complexity of flight networks:** Airlines must plan for many flights operating daily over multiple crew bases. A robust flight network must be generated and maintained by the airlines to assign crews to each flight effectively.
- **Flight planning:** Compared to cabin crew, cock-pit crew generation is more complex as pilots are qualified to fly specific aircraft. Efficient planning of crew pairings over several flight legs must account for pilot specifications while covering as many flights as possible in the network.
- **Risk mitigation:** Several factors could affect the smooth flow of daily flight operations (e.g., Severe weather, flight cancellations, etc.). Contingency plans for crew members must be kept in place in case of flight disruptions.
- **Regulatory constraints:** Regulatory constraints are set up by governing bodies and vary from region to region and keep the work assigned to crew members in check. These regulations include the number of flying hours, number of flights assigned and minimum transit/rest time between flights.

The crew pairing problem aims to cover all flight legs in the airline's flight schedule while satisfying all legal, regulatory and operational constraints and keeping the cost associated with constructing the pairings to a minimum. The crew pairings generated must be in accordance with the policies set up by airline governing bodies regarding crew working hours, rest/vacation time, compensation and penalties.

4.
Problem Formulation

The airline crew pairing problem uses the airline's flight schedule to generate an optimal list of pairings while accounting for the required constraints. It is traditionally formulated as either a Set Partitioning or a Set Cover Problem [3]. In the case of set partitioning, each flight leg is covered only once, i.e., there are no deadhead flights. On the other hand, a set cover problem allows for flight legs to be covered in more than one pairing, where the over-coverage represents deadhead flights. While a set partitioning model might yield a better solution, it may also produce infeasible solutions in problems like ours where an ideally portioned solution does not exist.

On the other hand, a set covering model could lead to faster convergence in the case of large-scale crew pairing optimization problems [3]. Our solution has modeled the problem as a Set Cover Problem. The following section describes our approach to problem formulation.

4.1 Decision Variables

The decision variables in this problem are binary variables, which indicate the selection of a particular pairing from a set of legal pairings, i.e., if the decision variable x_j has a value of 1, it represents that the j^{th} pairing is selected. The number of decision variables equals the number of pairings *(k)*.

4.2 Objective

This paper considers two objectives to optimize as described below. Each objective has a weight assigned to it. Setting either weight as zero formulates the problem as a single objective problem.

- **Pairing cost optimization:** The total cost incurred by the airline due to the allocation of different pairings has been minimized.
- **Deadhead minimization:** The total cost incurred by the airline due to deadhead allocations of flight legs should be minimized.

$$Min\left(W_p\sum_{j=1}^{P}c_jx_j + w_{dh}\left(\sum_{i=1}^{F}\left(\sum_{j=1}^{P}a_ix_j - 1\right)\times P_{Dhd}\right)\right)$$

$$P = number\ of\ Pairings$$
$$F = number\ of\ Flight\ Legs$$
$$c_j = cost\ of\ pairing\ p_j$$
$$x_j = \begin{cases}1, if\ pairing\ p_j\ is\ selected\\0, otherwise\end{cases}$$
$$P_{Dhd} = penalty\ for\ the\ number\ of\ deadheads$$
$$a_{ij} = \begin{cases}1, if\ flight\ f_i\ \ is\ covered\ in\ pairing\ p_j\\0, otherwise\end{cases}$$
$$w_p = weight\ for\ pairing\ objective$$
$$w_{dh} = weight\ for\ deadhead\ objective$$

4.3 Constraints

The pairings created in accordance with the governing policies are known as legal pairings and are based on several constraints:

- **Common airport between legs:** Keeping the arrival airport of the previous leg and the departure airport of the next leg the same allows for the smooth assignment of crews to their flights and reduces time delays and overall cost of the crew pairing.
- **Rest time:** The minimum and maximum rest time between flight legs for crew members, including the layover, transit or normal rest time between flights.
- **Common base for a pairing:** The start and end airport of a pairing should be the same. This airport should be one of the airline's bases, which is a subset of all the airports that the airline services. This ensures the airline does not incur additional costs in transporting the crew from or to the assigned pairing.
- **Flying hours:** The number of in-flight duty hours for cabin and cockpit members is strictly maintained due to the concentration required during the duty periods.

- **Number of flights:** Crew members could be part of flights either as part of their duty periods or as deadhead flights while moving to their destination. In accordance with airline regulations, crew members must be given a specific number of flights to be a part of and compensated as such.
- **Leg coverage:** All flight legs that the airline operates should be covered by at least one selected pairing.

$$\sum_{j=1}^{P} a_{ij} x_j \geq 1, \forall_i \in \{1, 2, ..., F\}$$

5.
Solution Methodology

Our current work explores the application of quantum annealer-based optimization for crew rostering along with a comparison to an open-source Python-based classical optimizer.

Solving any optimization problem requires two steps - one, formulating the problem, and two, obtaining the optimal solution to the formulation. The first step constitutes understanding the problem and formulating it in mathematical terms. This mathematical formulation can be done in several ways, such as Linear Programming (LP), Mixed Integer Linear Programming (MILP), non-linear and quadratic. Based on the convenience of formulating the problem and the availability of algorithms, techniques and tools to solve the model, a method of formulation is chosen. Once the problem is formulated, the second step in the optimization problem is obtaining the solution with the most optimal cost. As the problem size increases, the problem cannot be solved analytically and brute force methods theoretically could take exponentially longer. Hence, numerical approaches are utilized to provide approximate solutions to large optimization problems.

In the context of the crew pairing optimization problem, two approaches are typically followed. The first approach generates a set of legal pairings deterministically, followed by selecting the optimal subset of those legal pairings in the optimization step. The second approach combines the steps to dynamically generate pairings within the optimization step itself [4].

The second approach requires an assumption to be made about the number of pairings generated. It also considers all combinations of the set of flight legs which would cause the number of decision variables to exceed the capacity of current computing technology [5]. A separate legal pairing generation approach alleviates this problem.

Moreover, the legality of a pairing is defined by Federal Authority rules and regulations, labor agreements & government safety regulations. Such constraints cannot be violated for a penalty, which would typically be the case in penalty models such as Quadratic Unconstrained Binary Optimization models.

Hence, the first approach ensures that such constraints are satisfied while generating the set of legal pairings, while the optimization step focuses on the business objectives and constraints of the airline. For these reasons, we have followed the first approach in this paper.

5.1 Flight Network Generation

The flight schedule is provided with the following information:

1. Flight leg number
2. Arrival and departure airports
3. Date of departure and arrival
4. Estimated departure and arrival time

The first step is to create a network of connected feasible flight legs. The connectivity is based on whether the consecutive flight legs satisfy two of the constraints explained in section 4.3.

- Departure airport for the next leg should be the same as the arrival airport of the previous leg
- Inter-flight rest time should be within the specified range

This can be visualized as a pairing tree, as illustrated in Figure 1. Each root node corresponds to a flight leg with the edges connecting it to feasible succeeding flight legs. The process is repeated for all available flight legs in the schedule, creating a complete flight network covering all possible connections.

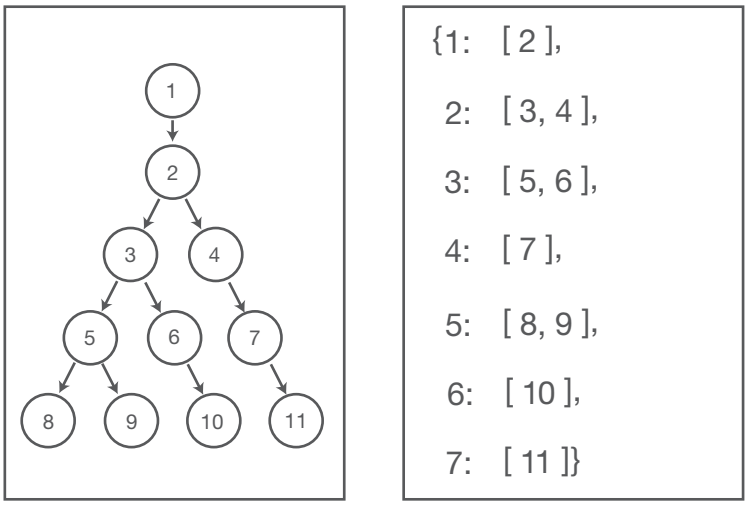

Figure 1: Pairing tree

5.2 Traversing the Flight Network

After the generation of the flight network, the next step is to create a set of legal pairings compliant with the next set of constraints. Using Depth-First Search, we traverse through the network, taking one root node at a time, and check if the flight sequence satisfies the following constraints:

- Flying hours for a pairing must be within a specified range
- The number of flight legs in a pairing must be within a specified range
- The arrival airport of the last node in the pairing should be the same as the departure airport of the first/root node

Due to the size of the flight network, it is not possible to generate a feasible solution (a set of pairings that covers all flights at least once) in a brief time using the original DFS. Hence, it is enhanced by varying the step length of backtracking to cover more unique flight nodes.

[4] Figure 2 shows the difference in unique flight nodes covered without backtracking vs. when backtracking with a step length of two. Apart from this, backtracking also reduces the overall computation time of graph traversal.

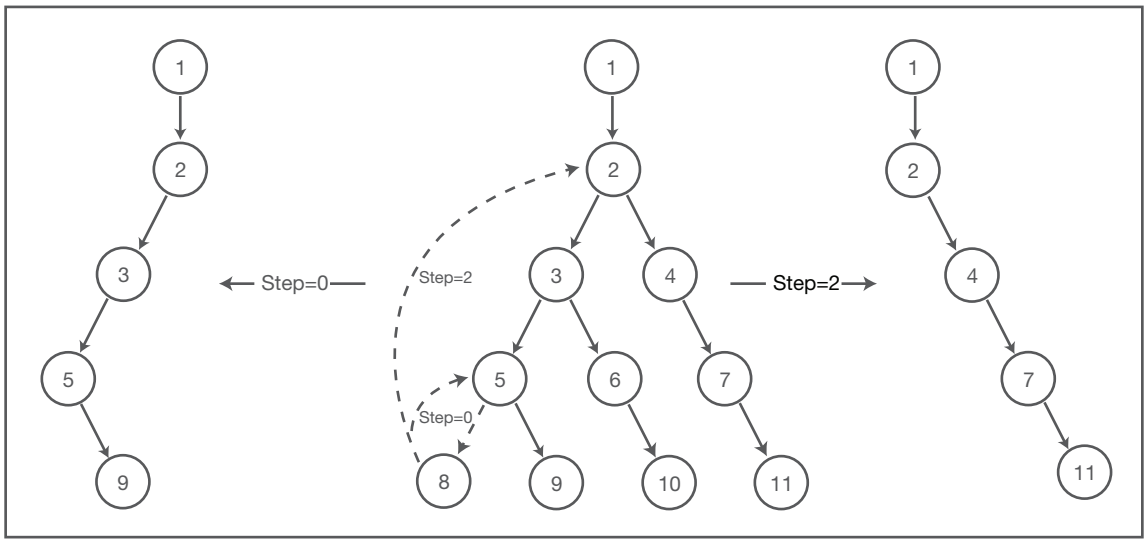

Figure 2: Depth first search with backtracking

Since the traversal of the graph with different starting nodes is independent of each other, multiprocessing can be used to decrease the computational time further. The overall pairing generation process is decomposed into independent subprocesses based on starting flight nodes. Each process returns a set of legal crew pairings starting from a given flight node. By running *'n'* independent processes in parallel, the legal crew pairing generation process is significantly sped up. [4] Further, since the constraints demand that the start and end of the pairing should be one of the airline's base airports, flights departing from a non-base airport need not be considered as a root node for traversal.

5.3 Optimization Using Set Cover Problem

Once the set of legal pairings is created, a Set Cover Problem can be formulated, as explained in section 4. In this step, we select the optimal number of legal pairings that cover all flight legs. Since the legality of pairings has already been enforced, the optimization problem has a more efficient search space, reducing the number of binary variables required and, subsequently, the computational time. In this paper, we take three approaches to optimization as described below.

5.3.1 Classical Optimization-based Approach

We have formulated the problem as a binary linear programming problem. Classical LP solvers have been used for decades to solve optimization problems for industry and academia. For the crew rostering problem, the open-source Python library 'Pyomo' has been utilized to model the optimization problem. Pyomo can be used to define abstract problems, create concrete problem instances and solve these instances with standard classical optimization routines. Pyomo provides a capability that is commonly associated with algebraic modeling languages like AMPL and GAMS. Pyomo leverages the powers of the Coopr software, which integrates Python packages for defining

optimizers, modeling optimization applications and managing computational experiments. COIN-OR Branch and Cut (CBC) solver has been used to solve the problem. It is an open-source mixed-integer linear programming solver.

5.3.2 Quantum-inspired Optimization Approach

The problem is formulated as a Binary Quadratic Model (BQM) using the Dimod package from D-wave's Ocean SDK. Simulated Annealing is used to solve the problem. D-wave's Simulated Annealing Sampler (SAS) is commonly used in heuristic optimization problems and approximate *Boltzmann* sampling, well suited to finding solutions for large problems. SAS approaches the equilibrium distribution by performing iterative updates at a sequence of decreasing temperatures, terminating at a target value **β**. Each spin is updated once in a fixed order per **β**.

Boltzmann distribution: $\frac{1}{\mathrm{k}_B \mathrm{T}} = \boldsymbol{\beta}$ k_B : Boltzmann constrant

T = Thermodynamic temperature (kelvin)

The Pyomo Formulation and CBC solver took exceedingly substantial time in our experimentation. To solve the multi-objective problem, CBC needed help finding a solution within 2 hours on the smaller datasets, while Pyomo faced a similar issue in problem formulation with increasing problem sizes. Hence, in the multi-objective problem, SAS has been employed to find solutions within a feasible time.

5.3.3 Quantum Annealer-based Approach

5.3.3.1 Why Quantum

Quantum annealers are essentially ISING machines that help solve combinatorial optimization problems. Solving optimization problems with quantum annealers requires encoding industry and academia problems to energy minimization problems. Quantum annealers employ energy encoding to map problems to hardware and follow a nature-inspired quantum optimization paradigm. It allows the system to evolve through time while maintaining control over the pace of evolution, and when given enough time, a system will achieve its lowest energy point.

While classical algorithms such as Simulated Annealing also employ a similar phenomenon, quantum annealers can improve performance and quality over such classical algorithms using quantum mechanical phenomena like quantum tunneling.

In the context of crew pairing, the complexity of problems increases with an increase in the number of flights and number of airline bases. Current classical optimization systems have been developed over decades and provide good approximations for medium-sized problems, but the run time increases with an increase in problem size. Quantum solvers such as hybrid classical-quantum solvers on quantum annealers and quantum-inspired classical optimization algorithms (QIO) can improve solution quality while reducing the run time for certain types of energy landscapes representing a particular class of problems.

5.3.3.2 Approach

D-wave's quantum annealers currently support optimization models in the form of Constrained Quadratic Models (CQM) or Binary Quadratic Models (BQM) to define objectives and constraints. BQMs (Binary Quadratic Models) are further transformed into Quadratic Unconstrained Binary Optimization (QUBO) or equivalent ISING formulation in ferromagnetism.

To use quantum annealers, the optimization problem must first be converted to CQMs (Constrained Quadratic Models) or BQMs. CQMs, as the name suggests, are constrained models with binary or integer decision variables. BQMs have only binary decision variables, and the constraints must be converted into an unconstrained problem using the penalty method. In this method, the constraints are added to the objective function with a penalty. If a solution fails to satisfy a constraint, the corresponding penalty will be added to the total cost.

Dimod has been utilized for the formulation of the CQM model. D-wave's Leap Hybrid Solver has been used to solve the problem. It implements state-of-the-art classical algorithms and intelligent allocation of the quantum computer to parts of the problem where it benefits most.

6.
Results & Discussion

6.1 Data Description

Table 1: Dataset description

Attributes			DS1	DS2	DS3	DS4	DS5	DS6	DS7	DS8
Dataset	Number of legal pairings		1013	1500	1013	1500	1500	1013	5613	5886
	Initial number of legal pairings		2824	4099	5742	8483	11630	13405	103440	148460
Constraints	Inter-flight rest time	Min	45 mins							
		Max	2 days							
	Flying hours	Min	1							
		Max	30							
	Number of flights	Min	2							
		Max	14							

6.2 Solution Quality

The following table details the comparative results of the different approaches on a mid-sized problem (DS4) with 1500 flights and 8483 pairings.

Table 2: Solution quality KPIs (Key Performance Indicators) for DS4

Attributes		Single Objective (100% Pairing)			Single Objective (100% Deadhead)			Multi-objective (70% Pairing + 30% Deadhead)		
Solver		**CQM Hybrid**	**BQM SAS**	**Pyomo + CBC**	**CQM Hybrid**	**BQM SAS**	**Pyomo + CBC**	**CQM Hybrid**	**BQM SAS**	**Pyomo + CBC***
Number of Pairings		363	639	410	557	645	580	402	695	No solution
Number of Deadhead Flights		330	500	6	67	488	8	128	465	
Number of Flights Missed		0	1	0	0	2	0	0	4	
Formulation Time(s)		1.15	4.17	1.54	33.29	40.88	0.472	31.91	39.56	
Solver Time(s)		5	122.81	5328	13.95	100.71	20.37	13.95	106.59	
Total Flying Hours	**Min**	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37	
	Max	21.18	23.98	22.46	25.7	24.9	20.3	24.02	23.98	
	Mean	6.17	3.8	5.46	3.41	3.75	3.14	4.92	3.42	
	Std. Dev	6	3.98	8	3.46	4.07	2.92	5.25	3.47	
Total Inter-flight Rest Minutes	**Min**	46	46	46	46	46	46	46	46	
	Max	14908	12651	12076	12584	12584	11388	13409	13468	
	Mean	2360.63	1503.93	988.58	1266.56	1517.55	855.41	1730.27	1400.51	
	Std. Dev	2872.54	1728.93	1278.42	1535	1877.36	1336.21	2271.42	1607.21	

***Note:** In Multi-objective problems, CBC could not find a solution within 2 hours of execution.

6.3 Performance Results

For the other datasets, we provide only performance KPIs comparing the different approaches.

Table 3: Performance KPIs across datasets

			Number of Pairings	Number of Deadheads	Flights Missed	Formulation Time(s)	Solver Time(s)
DS1	Pairing Only	CQM	170	257	0	0.3	5
		SAS	402	408	4	0.59	14.46
		CBC	164	6	0	0.473	4176
	Deadhead Only	CQM	323	64	0	2.06	5
		SAS	438	391	4	2.61	13.19
		CBC	306	7	0	0.124	9.89
	Multi Objective*	CQM	266	102	0	2	5
		SAS	409	417	4	2.55	12.74
DS2	Pairing Only	CQM	326	361	0	0.53	5
		SAS	607	473	1	1.11	33.65
		CBC	482	8	0	0.462	4824
	Deadhead Only	CQM	488	93	0	5.76	5
		SAS	577	472	1	7.09	27.09
		CBC	555	8	0	0.209	13.21
	Multi Objective*	CQM	399	137	0	5.45	5
		SAS	636	518	2	6.76	27.01
DS3	Pairing Only	CQM	200	283	0	0.72	5
		SAS	409	399	2	1.99	63.27
		CBC	220	8	0	0.576	4752
	Deadhead Only	CQM	333	52	0	15.15	6.44
		SAS	436	392	4	18.46	51.1
		CBC	255	5	0	0.403	28.856
	Multi Objective*	CQM	249	74	0	14.44	6.44
		SAS	411	433	2	17.73	53.35
DS5	Pairing Only	CQM	292	490	0	1.64	5
		SAS	566	501	4	8.01	221.31
		CBC	304	6	0	1.52	5832
	Deadhead Only	CQM	510	70	0	61.6	26.27
		SAS	631	541	5	76.62	180.06
		CBC	519	7	0	0.763	28.47
	Multi Objective*	CQM	344	182	0	59.42	26.27
		SAS	647	523	4	74.57	189.21
DS6	Pairing Only	CQM	232	276	0	1.78	5
		SAS	420	361	6	17.23	340.01
		CBC	276	8	0	1.32	6436
	Deadhead Only	CQM	339	57	0	116.72	47.89
		SAS	452	441	5	146.87	285.36
		CBC	280	5	0	0.73	95.62
	Multi Objective*	CQM	236	105	0	116.25	47.89
		SAS	426	410	7	146.13	299.67

DS7	**Pairing Only****	**CQM**	1917	444	0	2872.03	745.16
DS8	**Multi Objective****	**CQM**	2153	245	0	2420.70	639.52

***Note:** In Multi-objective problems, CBC could not find a solution within 2 hours of execution.
****Note:** In larger datasets, Pyomo was unable to formulate the model within 2.5 hours of execution, and Dimod was unable to convert CQM to BQM with the available RAM (30 GB).

6.4 Visualizations

This section illustrates the comparative performances detailed above. Figure 3 compares the time taken for the different solvers to find an optimal solution. Figure 4 compares the number of pairings selected by the solvers, whereas Figure 5 compares the number of deadheads.

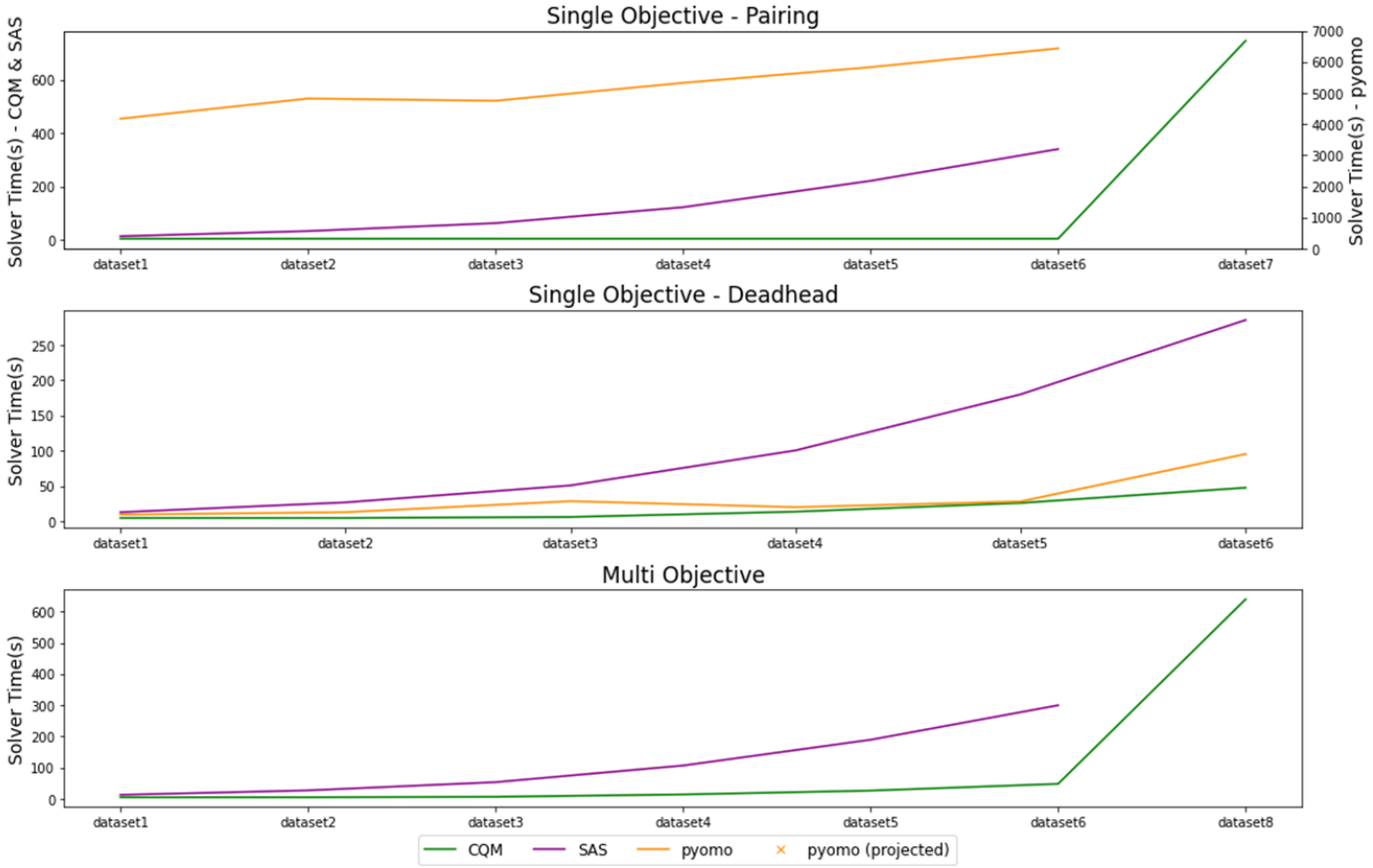

Figure 3: Solver time (in seconds) by dataset

- Considering pairings as the only objective, the CQM solver consistently takes less time than SAS and CBC, with CBC taking 1000x the time taken by CQM
- In the case of deadheads as the only objective, the time taken by SAS increases significantly with increasing problem size, whereas CQM and CBC time does not increase much
- In Multi-objective, CBC is unable to find a solution on DS1-DS6 within 2 hours. For all objective implementations, as the problem size increases (DS7 & DS8), Pyomo is unable to formulate the problem within 2.5 hours of execution, and Dimod is unable to create the BQM model within 2 hours of execution.

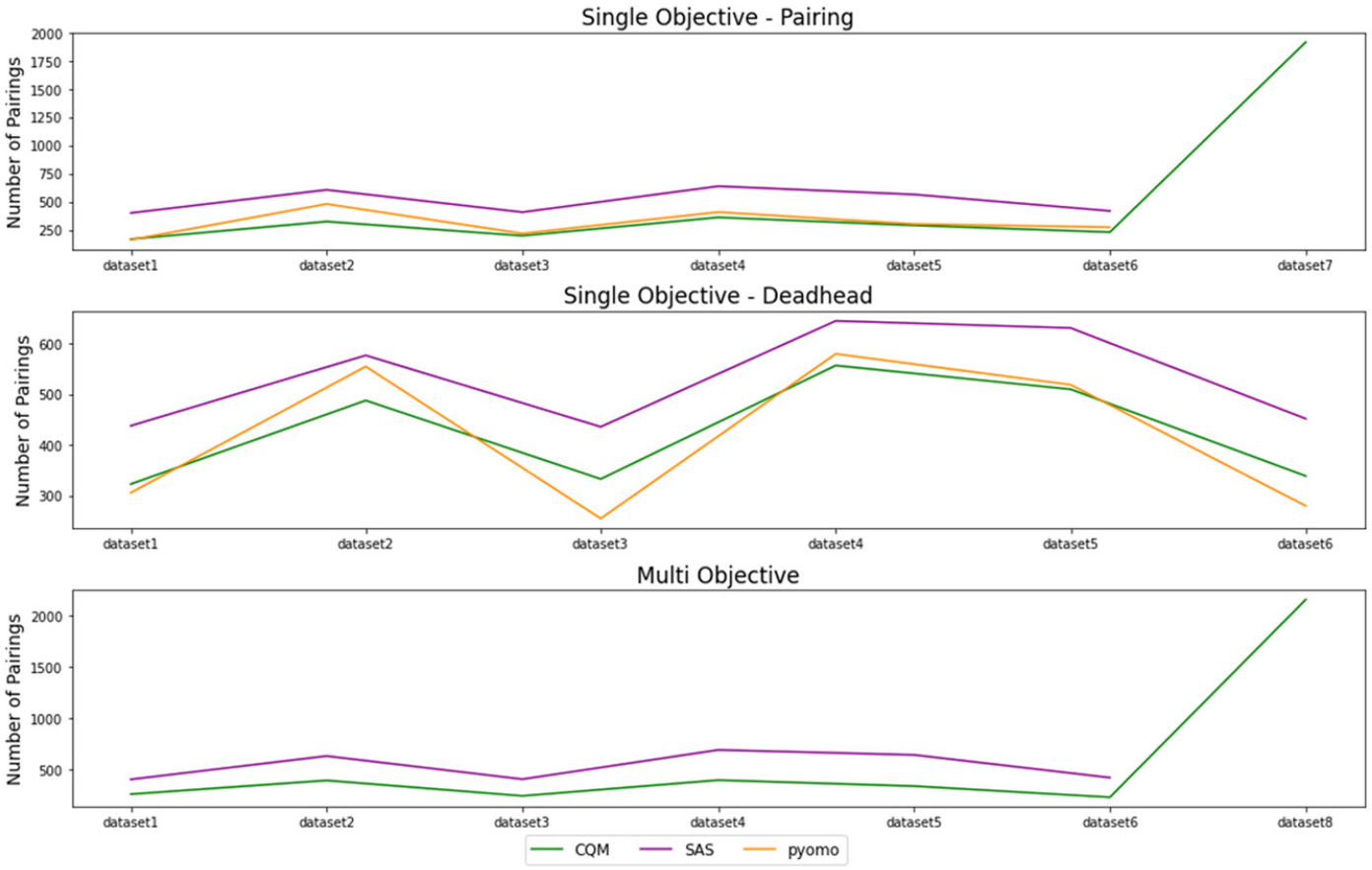

Figure 4: Number of pairings by dataset

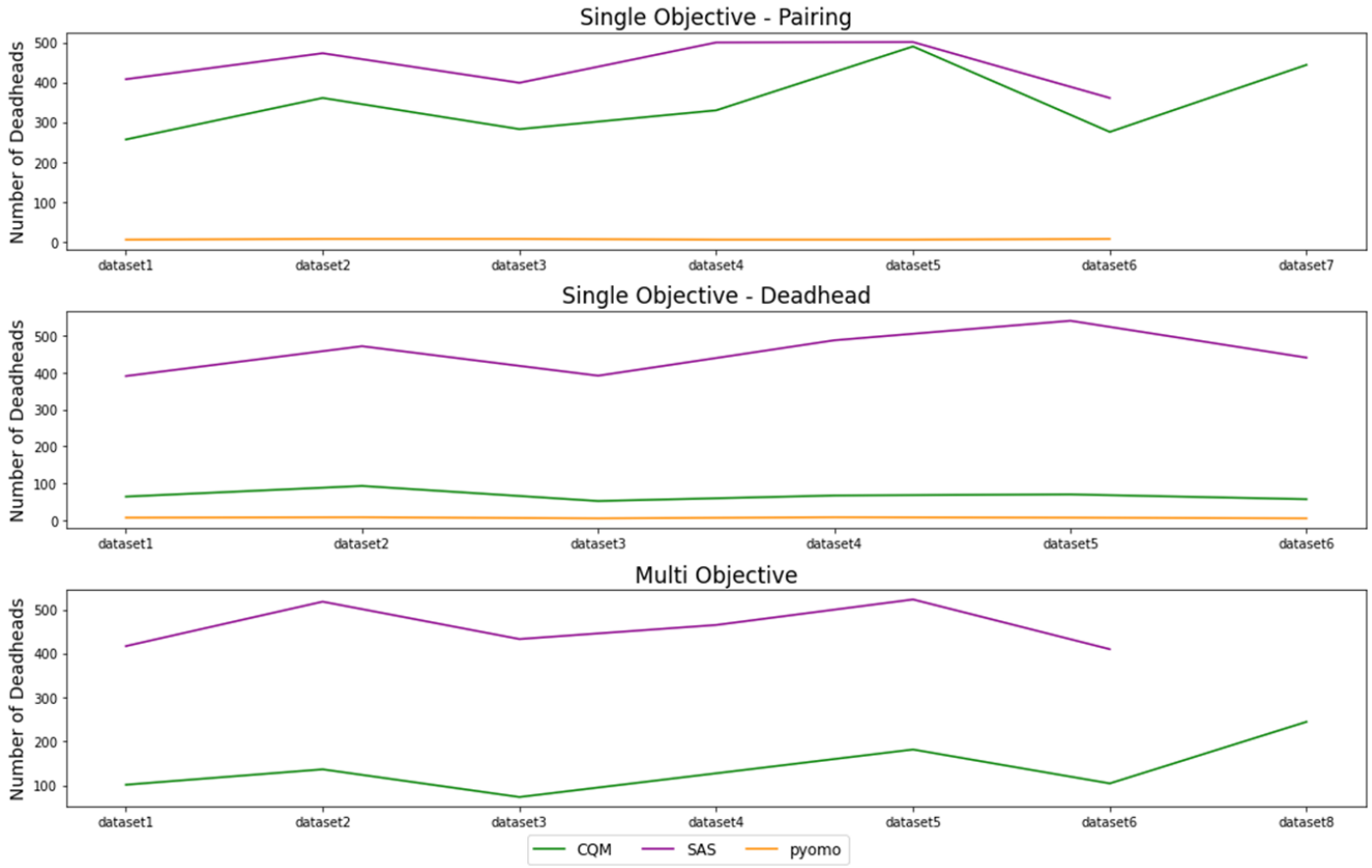

Figure 5: Number of deadheads by dataset

6.5 Analysis of Results

Quality of solution:

- For our experiments, we assess the quality of feasible solutions using the number of pairings and number of deadheads. Solutions not covering all flight legs are considered infeasible.
- While considering only pairings as a single objective, we can see from Table 2 that the best quality of solution is given by Pyomo & CBC; however, the solver takes over 1000x the time taken for the quantum solver. From the visualization, we can also see that the CQM solver takes the least amount of time across all datasets. Further, as the problem size increases, Pyomo is unable to formulate the model within 2.5 hours, whereas Dimod was unable to create a BQM model for SAS due to exhausting computational resources.
- With the deadhead-only objective, CBC and CQM were able to give feasible and comparable solutions. From Table 2, we can see that while CBC gave a lesser number of deadheads, CQM returned a lesser number of pairings and took less time to solve. Table 3 further supports the claim of comparative solution quality between CQM and CBC solvers.
- In Multi-objective problems, CBC could not find a solution within 2 hours of execution for the smallest dataset. As the problem size increased, Pyomo was unable to formulate the problem. Since SAS gave an infeasible solution and did not cover all flights, only CQM could handle the multi-objective problem and give feasible results.
- Across all implementations, SAS was unable to cover all flight legs, in every dataset while also returning more pairings and deadheads than the CQM and CBC solvers.

Scalability of solution: The CQM quantum annealer provides a scalable solution, as shown in Table 3. As the problem size increases, not only can the quantum solution return a valid solution within a feasible time, but after a specific problem size, Pyomo cannot formulate the multi-objective problem for CBC, and Dimod cannot create a BQM model for SAS. Further, as evident from Figure 3, CQM provides a consistently lower time to solution across all implementations compared to SAS and CBC solvers.

7.

References

[1] https://financesonline.com/number-of-flights-worldwide/ – Number of Flights Worldwide in 2022/2023: Passenger Traffic, Behaviors, and Revenue

[2 https://www.id1.de/2021/10/03/airline-statistic-how-many-daily-flights-do-airlines-operate/ - Airline Statistic: How Many Daily Flights Do Airlines Operate?

[3] https://arxiv.org/abs/2003.06423 - On Initializing Airline Crew Pairing Optimization for Large-scale Complex Flight Networks. Divyam Aggarwala, Dhish Kumar Saxenaa, Thomas Bäckb and Michael Emmerichb

[4] https://ieeexplore.ieee.org/abstract/document/8628699 - On Large-Scale Airline Crew Pairing Generation. Divyam Aggarwala, Dhish Kumar Saxenaa, Michael Emmerichb and Saaju Paulose.

[5] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3919d9221641d9a3eeaa975d8798d8a4b365c135 - Airlines' Crew Pairing Optimization: A Brief Review. Xugang Ye

About Mphasis

Mphasis' purpose is to be the *"Driver in the Driverless Car"* for Global Enterprises by applying next-generation design, architecture and engineering services, to deliver scalable and sustainable software and technology solutions. Customer centricity is foundational to Mphasis, and is reflected in the Mphasis' Front2Back™ Transformation approach. Front2Back™ uses the exponential power of cloud and cognitive to provide hyper-personalized ($C = X2C^{2}_{TM} = 1$) digital experience to clients and their end customers. Mphasis' Service Transformation approach helps 'shrink the core' through the application of digital technologies across legacy environments within an enterprise, enabling businesses to stay ahead in a changing world. Mphasis' core reference architectures and tools, speed and innovation with domain expertise and specialization, combined with an integrated sustainability and purpose-led approach across its operations and solutions are key to building strong relationships with marquee clients. Click here to know more. (BSE: 526299; NSE: MPHASIS)

For more information, contact: marketinginfo.m@mphasis.com

USA
Mphasis Corporation
41 Madison Avenue
35th Floor, New York
New York 10010, USA
Tel: +1 (212) 686 6655

UK
Mphasis UK Limited
1 Ropemaker Street, London
EC2Y 9HT, United Kingdom
T : +44 020 7153 1327

INDIA
Mphasis Limited
Bagmane World Technology Center
Marathahalli Ring Road
Doddanakundhi Village, Mahadevapura
Bangalore 560 048, India
Tel.: +91 80 3352 5000

NR 27/11/23 US LETTER BASIL8364

www.mphasis.com