Article

Optimizing Large-Scale Demand and Capacity Balancing in Air Traffic Flow Management Using Deep Neural Networks

**Abstract:** Over the past forty years, air traffic flow management (ATFM) has garnered significant attention since the initial approach was introduced to address single-airport ground delay issues. Traditional methods for solving both single- and multi-airport ground delay problems primarily rely on operations research techniques and are typically formulated as mixed-integer problems (MIP), with solvers employed to approximate optimal solutions. Despite their effectiveness in smaller-scale problems, these approaches struggle with the complexity and scalability required for large-scale, multi-sector ATFM, leading to suboptimal performance in real-time scenarios. To overcome these limitations, we propose a novel neural network-based Demand and Capacity Balancing (NN-DCB), which leverages neural branching and neural diving to efficiently solve the ATFM problem. Using data from 15,927 flight trajectories across 287 airspace sectors on a typical day in February 2024, our method re-allocates trajectory entry and exit times in each sector. The results demonstrate that large-scale ATFM problems can be solved within 15 minutes, offering a significant performance improvement over state-of-the-art methods. This study confirms that neural network-based approaches are more effective for large-scale ATFM problem-solving.

**Keywords:** Air Transportation; Air Traffic Flow Management; Demand and Capacity balancing; Airspace; Flight delay.

0. Introduction

Air traffic flow management (ATFM) aims to optimize air traffic flow through ground delay programs or rerouting strategies [1,2,3]. In recent years, many countries have progressively established central-distributed ATFM centers based on the evolved multi-airport ground delay theory [4,5]. In China, distributed ATFM systems have transitioned to centralized management systems that implement flow management strategies using uniform algorithms since 2021. However, from an engineering perspective, these systems face challenges such as low predictive accuracy, long computation times, and inefficiencies. These limitations hinder the effective implementation of ATFM in China. Furthermore, organizations such as the International Civil Aviation Organization (ICAO), Civil Aviation Administration of China(CAAC), Eurocontrol, Single European Sky ATM Research (SESAR), Federal Aviation Administration (FAA), and Civil Air Navigation Services Organisation (CANSO) have published future visions and roadmaps to guide air transportation towards higher safety, efficiency, and lower carbon emissions. With the increasing use of Artificial Intelligence (AI) techniques, the civil aviation industry, especially ATFM, is encouraged to adopt innovative methods [6].

The capacity-demand balancing problem is central to ATFM systems, where demand must not exceed capacity at any stage for all sectors. Multi-airport ground delay programs and demand-capacity balancing (DCB) in ATFM typically constitute mixed-integer problems (MIP) [1,7]. Historically, the consensus was to apply operations research techniques and solvers such as CPLEX [8] or Gurobi [9], and other popular solvers like CBC [10], Xpress [11] to approximate optimal solutions. to approximate optimal solutions. However, these solvers [8,9,10,11] often yield suboptimal solutions and are inefficient in some cases. To address these issues, this study develops a customized neural network designed to find optimal solutions within the national airspace. Recently, DeepMind [12] demonstrated success in rapidly solving MIP problems using neural networks, which manage the upper and lower bounds of traditional optimization problems. This motivates us to explore whether such techniques can be applied to solve DCB problems. The DCB method was first proposed by Xu et al. [13], who later developed a series of variant models [14,15]. Many scholars have adopted this approach to tackle ATFM challenges [16,17]. More recently, some researchers have proposed reinforcement learning-based methods, such as Q-learning, TD-learning, DQN, and A3C, to address the DCB problem [18]. These studies typically focus on using Ground Delay Programs (GDPs) to control flight operations, with real-world scenarios ranging from 400 to 12,000 flights. Additionally, some studies consider sector-opening schemes. Kravaris et al. [19] explore how multiagent reinforcement learning can address and mitigate imbalances between demand and capacity in air traffic management. Spatharis et al. [20] investigate collaborative multiagent reinforcement learning frameworks aimed at improving coordination and efficiency in air traffic management. Spatharis and Bastas et al. [21] propose hierarchical multiagent reinforcement learning strategies to optimize decision-making and management processes in complex air traffic scenarios. Mas-Pujol et al. [22] explore how image-based multi-agent reinforcement learning techniques can be applied to balance demand and capacity in complex systems like air traffic management. In summary, while existing methods[15,16,23,24,25,26,27] have alleviated the demand and capacity imbalances in ATFM to some extent, they still face challenges such as low efficiency, poor scalability, and insufficient real-time processing capabilities when dealing with large-scale, dynamically changing, complex airspace environments. To address these shortcomings, this study proposes a novel neural network-based demand and capacity balancing method (NN-DCB), which aims to provide a more efficient, flexible, and scalable solution by utilizing neural branching and neural diving techniques. The NN-DCB is the first to propose a customized neural network based on DeepMind's work, demonstrating that deep learning can be applied to solve MIP-based ATFM problems. Our primary contributions are as follows

* We successfully developed a deep neural network to obtain optimal solutions for large-scale ATFM problems, which address demand-and-capacity balancing within the ATFM framework;
* Our experiments focus on the complexity of real-world operational scenarios. We sourced data from Variflights and other reliable platforms, using large-scale flight trajectory data to train the model and ensure robust outcomes;
* Simulation results demonstrate that our method achieves the shortest solution time for real-time ATFM problems. The algorithms can determine the optimal delay times for each aircraft within 15 minutes, which holds significant practical value for real-world applications.

The structure of this paper is as follows. Chapter 0 introduces the background, methodology, and key contributions. Chapter 1 presents the deep learning and neural network methods employed. Chapter 2 outlines our proposed NN-DCB model, incorporating neural network theory, with a focus on the model’s objectives and constraints. Chapter 3 includes a case study based on our data and experimental results, along with a comparative analysis of traditional methods. Finally, the conclusion summarizes our findings and suggests directions for future research.

1. Related Work

1.1. Mixed-integer Problems Based on Deep Learning

Recently, the joint research team of DeepMind and Google published their work on designing a heuristic algorithm to solve large-scale mixed-integer programming (MIP) problems using shared structures from large datasets. Their heuristic algorithm generates high-quality joint variable assignments, targeting the two core sub-problems of MIP [28]. This method reduces the gap between variable values and optimal values. Neural diving and neural branching, based on neural networks, are integrated with MIP solvers such as SCIP [29]. Specifically, neural diving trains a deep neural network to generate values for integer variables, while SCIP is used to handle the remaining variables for smaller MIPs, leading to high-quality joint variable assignments [30]. Neural branching, on the other hand, trains a deep neural network to select branching strategies in the branch-and-bound process. This approach, combined with techniques like Full Strong Branching, reduces the gap between target and optimal values by constructing smaller decision trees and generating new variables.

The research team evaluated the neural networks on multiple real-world datasets, including MIPLIB and Google's production datasets. After pre-solving, solutions were found for most cases involving between 103 and 106 variables and constraints, which is significantly larger than previous learning-based methods. They announced that their approach marks the first major advancement over existing methods and MIPLIB benchmarks.

1.2. Mixed-integer Problems Based on Neural Network

Mixed-integer programming (MIP) is an NP-hard optimization problem widely encountered in domains such as supply chain management, production scheduling, and energy systems. In recent years, neural network-based approaches [12,31] have emerged as promising tools for solving MIPs more efficiently by leveraging machine learning techniques. Traditional methods for solving MIPs, such as branch-and-bound or cutting planes, often struggle with the exponential growth in complexity as problem sizes increase. Neural networks, particularly deep learning models, have been applied to either enhance existing solvers or provide entirely new frameworks for tackling MIP problems. By learning from past problem instances, neural networks can predict optimal or near-optimal solutions, accelerate branching strategies, and improve variable selection in solvers, thereby significantly reducing computation time. These methods have been explored

**Table 1. Comparative of DCB Methods Incorporating Deep Learning and Reinforcement Learning Methods**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Methods** | **RL method** | **Deep learning** | **ATFM method** | **Sector opening scheme** | **Uncer-tainty** | **Fair-ness** | **Experimental Scenario** | | **Flight scale( 103 )** | **Sector scale** |
| Real World | Hotspots |
| Kravaris[24] | Q-learning | / | GDPs | × | × | × | × | N/A | 1 | 16 |
|  |
| Duong[26] | Q-learning | / | GDPs | × | × | × | ✓ | N/A | 0.4 | N/A |  |
|  |
| Agogino[23] | DQN | / | GDPs | × | × | × | × | 54 | 3 | 16 |  |
| Huang[27] | A3C | / | GDPs | × | × | × | ✓ | 31 | 8.2 | 356 |  |
| Agogino[23] | TD-learning | / | MIT | × | × | × | ✓ | N/A | 1.3 | N/A |  |
|  |
| Spatharis[25] | Q-learning | / | GDPs | ✓ | × | × | ✓ | 53 | 6 | 169 |  |
|  |
| Tang[15] | PPO | / | GDPs | × | × | × | ✓ | 31 | 8.2 | 356 |  |
| Chen[16] | DQN | / | GDPs | ✓ | × | × | ✓ | 186 | 12 | 396 |  |
| Ours | / | SCIP | GDPs | × | × | × | ✓ | 120 | 15.9 | 287 |  |
| Current related study summarize | Most study adopt RL method and the GDPs strategy/Most studies do not consider the effects of fairness and uncertainty/Most study focus on small flight scale and experimental scenarios are normally the real world. | | | | | | | | | | |

in various applications, including air traffic flow management, where MIP formulations are prevalent due to complex capacity and routing constraints. The integration of neural networks into MIP-solving pipelines has demonstrated considerable potential in improving both the speed and scalability of solving large-scale, real-world optimization problems.

To address these, we present a solver enhanced with neural heuristics, which learns from training datasets and adapts to the problem domain. Additionally, the solver can output a new timetable while adhering to capacity and operational constraints. Our approach integrates two neural network-based components in the MIP solver: neural diving and neural branching. Neural diving primarily addresses binary variable problems, while neural branching is more suited to complex constraint interaction problems. In this study, we employ both techniques in the MIP solver, combining them to produce a customized neural solver for the Demand-Capacity Balancing (DCB) problem. Further details can be found in Figure 1 and Figure 2.

****

**Figure 1.** A Hybrid Neural Approach for MIP Optimization in ATFM. Our approach integrates Neural Diving and Neural Branching within a customized Neural MIP Solver to address demand and capacity balancing in air traffic management. By combining generative modeling and imitation learning, the framework efficiently handles complex real-world MIP scenarios, ensuring optimal solution assignments for high-demand airspace sectors.

****

**Figure 2.** Bipartite Graph Representation for Neural Network-Based MIP Solving. This image illustrates a bipartite graph representation of a mixed-integer programming (MIP) problem used as input for a neural network.

2. Methodology

2.1. Overview

Building on the use of neural networks to solve MIP problems, we introduce our Demand and Capacity Balancing method, based on neural branching and neural diving (NN-DCB). The NN-DCB model is an extension of the DCB model, an optimization framework for demand-and-capacity balancing. The primary goal of this problem is to minimize flight delays while adhering to capacity and other operational constraints. The DCB model was first introduced in Xu’s paper [13], with further models presented in subsequent works [14,15]. The initial model focused solely on flight delays and classic ATFM constraints. In later works, additional strategies and objectives were introduced under various scenarios. The C-DCB variant incorporates rerouting strategies and fuel consumption across different air routes, while SC-DCB1 [13] and SC-DCB2 [15] add more operational constraints, making the model more suitable for real-world scenarios. To compare the methods more intuitively, we compare several representative methods in Table 1.

However, none of these works have yet addressed large-scale air traffic flow management problems using novel neural networks. In this chapter, we present the NN-DCB model to make the theory more applicable to large-scale ATFM problems. The descriptions of some notations are listed in Table 2.

**Table 2.** Notations used in this paper and their descriptions

|  |  |  |
| --- | --- | --- |
| **No.** | **Notations** | **Descriptions** |
| 1 |  | set of flights |
| 2 |  | set of elementary sectors |
| 3 |  | set of time moments |
| 4 |  | set of time periods |
| 5 |  | set of operating sectors |
| 6 |  | subset of elementary sectors flight traverses |
| 7 |  | subset of time feasible for flight entering elementary sector |
| 8 |  | subset of time moments subject to time period |
| 9 |  | subset of operating sectors opened in time period |
| 10 |  | subset of operating sectors constructed by elementary sector |
| 11 |  | 1st elementary sector for flight that functions in operating  sector in time period | |
| 12 |  | elementary sector for flight |
| 13 |  | scheduled flight time of segment for flight |
| 14 |  | capacity of operating sector during time period |

2.2. NN-DCB

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

Although most ATFM models account for both ground and airborne delays, this paper focuses solely on ground delay to simplify the problem. The objective function represents the total deviation in time for flights passing through all airspace sectors. In Formula 1, denotes the actual flight time for each sector, which is the target of our optimization model, while represents the scheduled time for flight passing through sector . Thus captures the time deviation or ground delay. The assignment decision variable is denoted by , as shown in Formula 2.

**Constraint for operational limitations:** We define the time window as the variable in this paper, representing the maximum allowable delay for a flight, which enables us to assign delays accordingly. For example, if is set to 360 minutes, the maximum delay for flight is 360 minutes. Formula 3 represents the location constraint for flight . In this context, indicates that the flight has moved to a different location or sector. Formula 4 ensures that only one flight occupies a given sector at any time. Formula 5 guarantees that only one trajectory is selected.

**Constraint for airspace capacity:** We define airspace capacity as . Therefore, at any given time, the number of flights cannot exceed this airspace capacity, as expressed in Formula 6. In China, airspace capacity is determined within a 15-minute time window. Every 15 minutes, the CAAC authorities announce the updated capacity for each sector. We use to represent the constraint that flight flow cannot exceed capacity during each 15-minute interval. It is worth noting that this differs slightly from the approach in the other DCB model, where the capacity in France is updated every 60 minutes. Compared to the 15-minute window, this results in longer time intervals. For the case study in this paper is set to 15 minutes.

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |
|  | (5) |
|  | (6) |
|  | (7) |

**Constraint on airspace capacity:** Finally, the model must satisfy the binary constraint, where the decision variable is defined as either 0 or 1, as shown in Formula 7.

2.3. Training Details

|  |
| --- |
| **Algorithm 1** Learning to solve DCB problems. |
| 1: Initialize best schedule  2: Initialize best delay  3: **for** = 1 to **do**:  4:  5: // Neural Diving Phase  6: **for** in **do**:  7: delay\_prediction = ND ()  8: Explore delay: delay = delay\_prediction + (1-) random\_delay()  9: Apply delay: delay  10: **end for**  11: // Neural Branching Phase  12: **for** in **do**:  13: branch\_decision = NB ()  14: **if** branch\_decision[0] > 0.5 **then**  15: = small\_delay  16: **end if**  17: **if** branch\_decision[1] > 0.5 **then**  18: = small\_delay  19: **end if**  20: **end for**  21: // Constraint Check and Best Schedule Update  22: **if** check\_constraints (, C) **then**  23: calculate\_average\_delay(, D)  24: **if** **then**  25:  26:  27: **end if**  28: **end if**  30: Update based on optimization performance  31: **end for**  32: **Output**: Optimized schedule and best average delay |

2.3.1. Training Algorithm

To convert the classic model described in Section 1.1 into a neural network-based solution, we propose a novel neural network approach. As outlined earlier, our method applies machine learning to address two key subtasks of the MIP solver: (a) outputting variable assignments that satisfy the constraints (if such assignments exist); and (b) determining the range of the target value gap between the variable assignment and the optimal solution. These two tasks define the core components of our approach, namely Neural Diving and Neural Branching. Further details can be found in Figure 2.

**Neural Diving:** This component focuses on finding high-quality joint variable assignments.

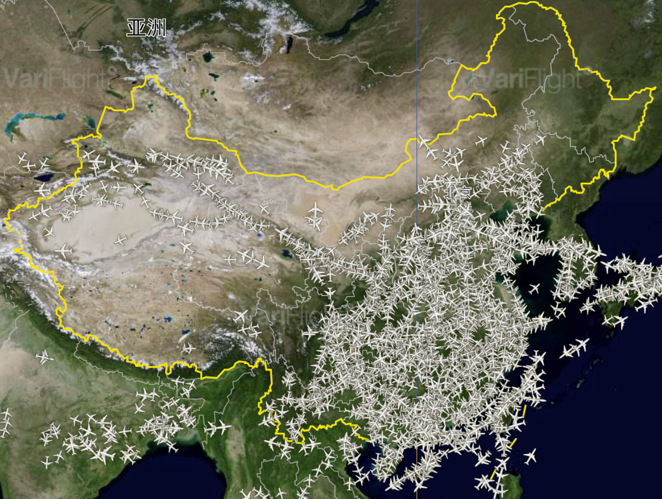
**Neural Branching:** This component is used to reduce the gap between the best assignment and the target value of the optimal solution.

Specifically, the Neural Diving model predicts continuous delay values for each flight, while the Neural Branching model makes binary decisions for minor adjustments. The optimization process combines both models to efficiently explore the solution space. Next, the method must satisfy the airspace capacity constraint. The constraint component ensures that the modified schedule adheres to time-varying sector capacities. The DCB learning algorithm is summarized in Algorithm 1. The first step involves constructing the Neural Diving model, followed by the construction of the Neural Branching model. After building the neural network architecture, the optimization process proceeds, aiming to solve the DCB problem by determining the optimal holding time for each flight and the appropriate takeoff time. The neural network architectures for both neural diving and branching consist of fully connected layers, designed specifically for Mixed-Integer Programming (MIP) optimization. The neural diving network uses 4 layers with ReLU activation in the hidden layers, with dropout (rate 0.3) applied for improved generalization and stability. The neural branching network has 3 layers with similar activations and a dropout rate. Both networks use Xavier initialization for weight initialization to prevent gradient issues. These design choices enable the networks to effectively explore and narrow the solution space for large-scale optimization in Air Traffic Flow Management. The Binary Cross-Entropy (BCE) loss for Neural Diving is defined as Formula 8. The BCE loss function is well-suited for binary decision-making tasks in Mixed-Integer Programming optimization, where the goal is to assign values of 0 or 1 to variables. While alternatives such as Huber loss were considered, they are more appropriate for regression tasks and mixed-variable problems, which would introduce unnecessary complexity in this binary context. For optimization, we employed the Adam optimizer, which is known for its adaptive learning rate and efficient handling of sparse gradients, using default parameters (β1 = 0.9, β2 = 0.999). Training adjustments included learning rate decay, starting at 1e-4 and halving every 10 epochs if the validation loss plateaued, and early stopping to prevent overfitting by halting training after 20 epochs without improvement.

|  |  |
| --- | --- |
|  | (8) |

Neural diving and branching methods distinguish themselves from traditional deep learning techniques by being specifically designed to solve Mixed-Integer Programming optimization problems, rather than general prediction or classification tasks. While traditional deep learning models are suited for sequential decision-making, they tend to be computationally expensive in large-scale optimization settings. Neural diving and branching methods focus on efficiently exploring and narrowing the solution space. Their direct integration with MIP solvers enhances solution quality and reduces computation time, making them highly effective for real-time optimization. In the context of Air Traffic Flow Management, which typically involves large-scale, dynamic problems with both discrete and continuous variables, neural diving and branching offer superior efficiency for handling extensive datasets. These methods are better suited for dynamic environments and can quickly adapt to changes, such as weather disruptions or fluctuating traffic. Consequently, they provide a powerful tool for optimizing demand and capacity balancing in ATFM, delivering faster and higher-quality solutions compared to traditional deep learning approaches.

2.3.2. Training Scenarios

****

**Figure 3.** Flight Distribution Across Chinese National Airspace.

The scenario data for this paper consists of nominal airspace capacity and flight information. We utilize real-world data from Chinese airspace, as shown in Figure 3. This image, provided by Variflight[[1]](#footnote-1), illustrates the distribution of flights across Chinese national airspace. The dense clustering of aircraft highlights high-traffic areas, predominantly in the eastern and southern regions of China, while the western regions see significantly fewer flights. A total of 15,927 flights were selected for training, using the sector opening scheme for that day within China's national airspace. Two hundred and eighty-seven sectors were considered, including airport terminal areas and en-route sectors. Based on the aforementioned real-world data, each training scenario includes a set of flight information and other essential details for the specific day.

3. Experimental

3.1. Data preprocessing

**Table 3.** Exhibition of Data Structure from Variflight.

|  |  |
| --- | --- |
| **Category** | **Data** |
| Flight Number | ZH9087 |
| Origin Airport | SZX |
| Destination Airport | HAN |
| Aircraft Number | B8079 |
| Flight type | A320-200 |
| Scheduled Departure | 2024/2/5 0:05 |
| Scheduled Arrival | 2024/2/5 1:00 |
| Actual Departure | 2024/2/5 0:23 |
| Actual Arrival | 2024/2/5 1:00 |
| Flight Time | 2024/2/5 0:13 |
| Speed | 12.96399975 |
| Vertical speed | -64 |
| Angle | 331 |
| Height | 0 |
| Longitude | 113.8068924 |
| Latitude | 22.64029503 |

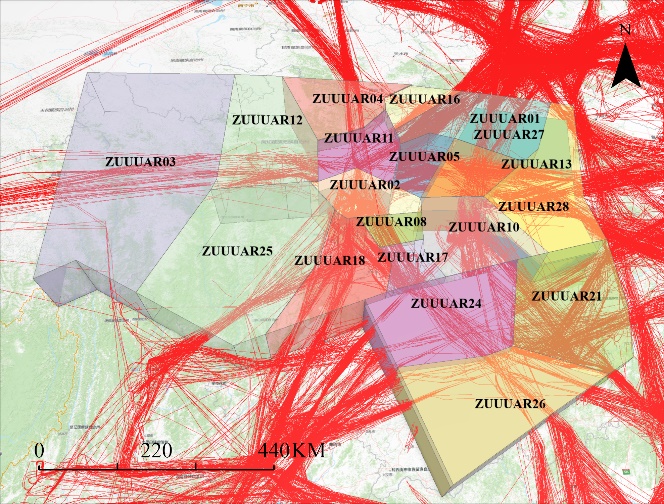
We retrieved the flight information dataset from Variflights. The original dataset contains 19,014,071 rows of data, including flight number, origin airport, destination airport, registration number, flight type, scheduled departure and arrival times, actual departure and arrival times, flight time, speed, angle, altitude, and longitude and latitude. The data was then cleaned, and relevant information was extracted from the Variflight dataset. Ultimately, 15,927 flights and their trajectories through 287 sectors were selected from the original dataset. The data represents a typical day (February 5th, 2024), summarizing the operations of 15,927 flights in Chinese national airspace.

We then retrieved the airspace information dataset from the Aeronautical Information Publication (AIP) provided by the Civil Aviation Administration of China (CAAC), which contains data on 287 sectors in Chinese national airspace. However, due to confidentiality issues, height information is only available for 285 sectors. The dataset includes details on altitude, longitude, and latitude, allowing us to model the airspace using ArcGIS software. After inputting the flight trajectories and airspace data, the software generated information on the air traffic situation. As shown in Figure 4, we simulated the complete operational scenario for China on February 5, 2024. The 15,927 trajectories traversed 287 sectors, as visualized in the ArcGIS maps. Figure 5 provides a close-up view, highlighting the large number of trajectories over 27 sectors within the Chengdu Shuangliu International Airport (IATA: CTU, ICAO: ZUUU).

Finally, we used the calculation function in ArcGIS to obtain the entry and exit times for all trajectories across 287 sectors. The processed dataset, consisting of 167,473 data points, provides detailed insight into the operational situation on that day. Using these entry and exit times, the optimal time can be calculated with our NN-DCB model and proposed algorithm.

****

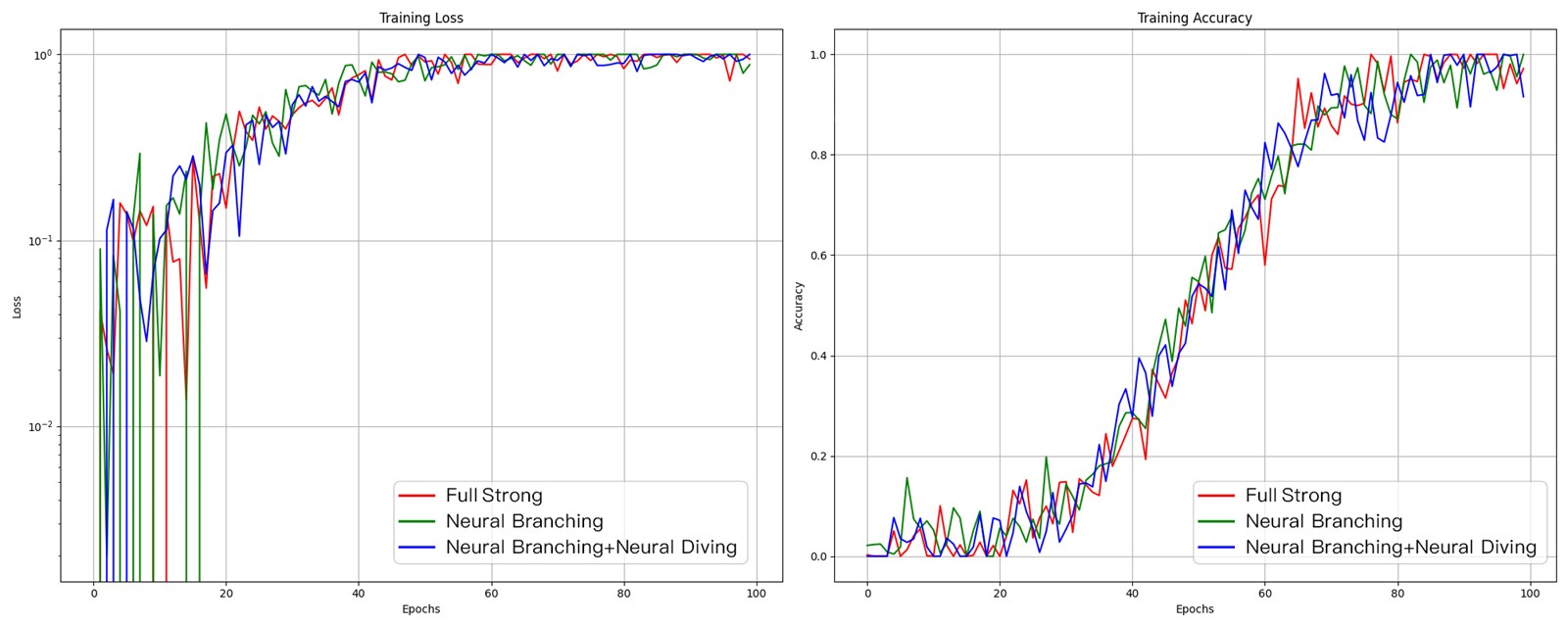
**Figure 4.** Real-Time Air Traffic Trajectories and Sector Distribution on February 5th, 2024, in Chinese National Airspace. The red lines represent flight trajectories, while the shaded regions indicate different airspace sectors, with altitude levels denoted by color gradients.

****

**Figure 5.** Zoomed View of Terminal Area Air Traffic Flow and Sectors. This image provides a zoomed-in view of the terminal area air traffic flow and sector distribution in Chengdu on February 5th, 2024. The red trajectories represent individual flight paths, while the colored regions denote different airspace sectors in the Chengdu ACC (ZUUU).

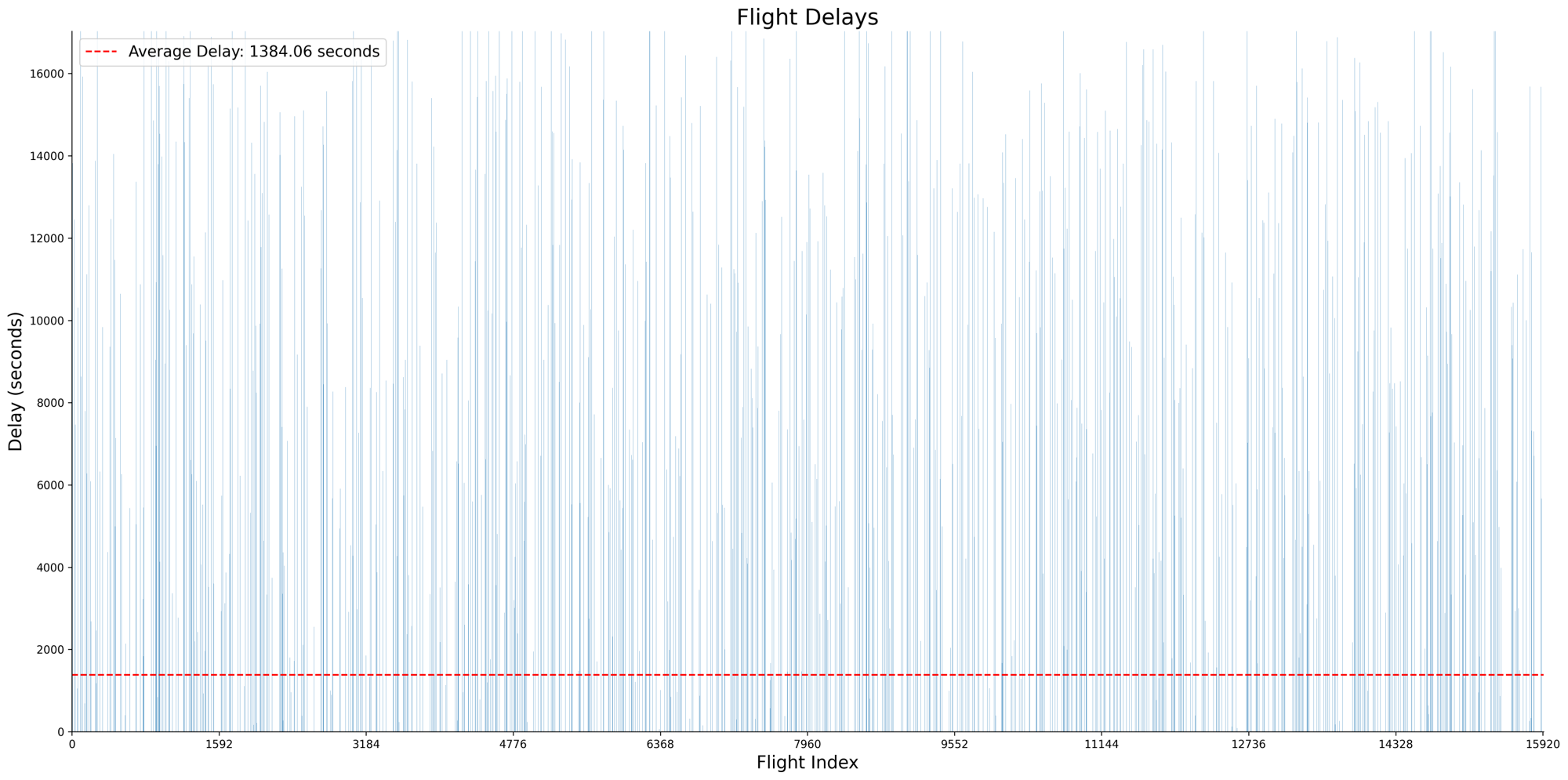
3.2. Experimental Results

The neural networks were constructed using the PyTorch library in a Python environment and trained on an Apple MacBook Air equipped with an M2 chip, featuring a CPU, GPU, and 8 GB of RAM. The total computation time was 15 minutes and 50 seconds to find the optimal solution. Given that our simulation operates at the national level, this computational efficiency is highly acceptable and faster than RL-based or traditional optimization methods.

****

**Figure 6.** Comparison of Training Loss and Accuracy. The left graph shows the training loss over 100 epochs, with all methods converging after initial fluctuations, achieving a low loss value. The right graph illustrates training accuracy, where all methods demonstrate a similar upward trend, stabilizing around 100 epochs with accuracy nearing 1.0.

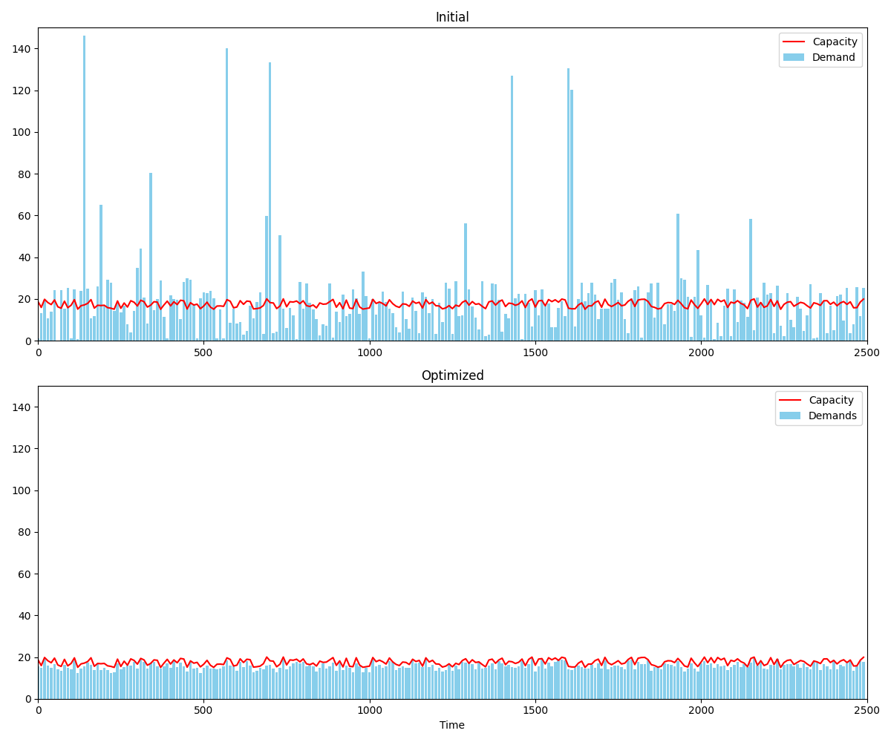
The training results are displayed in Figure 6. The left side of the figure shows the training loss, which measures the gap between the model's predictions and the actual values during training. Training loss is directly related to the optimization direction and the performance improvement of the model. The green curve represents the Neural Branching training loss, while the blue curve represents the combined Neural Branching and Neural Diving training loss. The trend of these curves is nearly identical: after a brief period of fluctuation, the loss rises sharply, stabilizes, and eventually converges. Both neural networks achieve minimal loss around 60 epochs. The right side of Figure 6 illustrates training accuracy, which refers to the model's accuracy on the training dataset. The trend of all three curves is similar, with training accuracy stabilizing around 100 epochs. In terms of performance, the results demonstrate that the combination of Neural Branching and Neural Diving achieves outstanding training performance.

****

**Figure 7.** Number of delay flights and average delay time. The y-axis represents the delay time in seconds, and the x-axis corresponds to individual flight numbers. Each blue bar indicates the delay duration of a specific flight, with a red dashed line marking the average delay of 1,386.56 seconds.

3.3. Flight Delays Distribution

Our approach aims to maximize computational efficiency while minimizing total delay time, leading to the generation of a new flight schedule. We obtained the flight delay distribution for 15,927 flights, as shown in Figure 7. The maximum delay time was approximately 18,365 seconds for flight HO1283, and the minimum delay was 17 seconds for flight CZ6044 on that day. A total of 2,428 flights were delayed in the simulation, accounting for about 15% of all flights. Additionally, the average delay time was 23 minutes, which is slightly higher than that in Tang [15] and Huang's [27] method. We attribute this difference to the larger sample size used in our study. Our experimental results confirm that the proposed method achieves better demand and capacity balancing, while satisfying the GDP strategy. As shown in Figure 8, before the simulation, demand in some sectors exceeded capacity. In certain circumstances, a large number of flights appeared in the same sector within a short period, severely impacting flight safety. After the simulation, most sectors achieved demand and capacity balance, with all demand curves falling below the capacity limit. These results are consistent with our previous work, with the key difference being the application of the neural network method to large-scale, complex operational scenarios, demonstrating higher computational efficiency and more reliable solutions.

****

**Figure 8.** Demand and Capacity Balancing within NN-DCB. The top graph shows the initial state, where the blue bars represent demand and the red line indicates sector capacity. Significant demand spikes can be observed. The bottom graph illustrates the optimized state, where demand has been adjusted to remain consistently below capacity across all sectors.

3.4. Comparison with State-Of-Art RL-based and Traditional DCB methods

This test aims to compare the proposed method with existing RL-based and traditional DCB methods. Specifically, we compare our method with those proposed by Chen and Xu. In Chen's study [12], he introduces a MARL (Multi-Agent Reinforcement Learning) method that considers the sector opening scheme. His study designs a multi-iteration mechanism within the DCB decision-making framework to address issues arising from non-stationarity in MARL and to ensure the elimination of all hotspots. In Xu’s study [10], two variant models are proposed to manage the sector opening scheme, using a traditional optimization method to find the optimal solution. In this section, we use "Xu-SC-DCB" to refer to Xu’s work and "Chen" to refer to Chen’s work.

|  |  |
| --- | --- |
|  |  |
| (**a**) Computing Time | (**b**) Flight Scale |

**Figure 9.** Computing Time and Flight Scale Comparison of Various Demand-Capacity Balancing Methods.

As shown in Figure 9, NN-DCB refers to the method proposed in this work, while the state-of-the-art RL-based methods solve the DCB problem faster than the traditional DCB methods used in Xu’s paper, our approach finds the optimal solution in just 15 minutes, outperforming even the RL-based methods. Furthermore, our simulation considers 15,927 flights, significantly larger than the datasets used in Xu's and Chen’s studies. Specifically, our flight scale is approximately 178% and 132% larger than those in Xu’s and Chen’s work, respectively. Despite the larger flight scale, our computing time is reduced to 53.6%, 29.4%, and 42.6% of the computing times in Xu's, Chen's, and RL-based methods, respectively. These results demonstrate that the proposed method outperforms both state-of-the-art RL-based and traditional methods. In terms of the decision-making framework, all methods are similar in that they consider synchronous demand-capacity balancing in collaborative air traffic flow management. In addition, the proposed NN-DCB is highly scalable, based on a standard framework design, and can be efficiently deployed on industry-standard computational hardware, including high-performance servers, GPUs, and cloud-based systems. This enables fine-grained, real-time air traffic flow management and trajectory coordination, even in large-scale, dynamic environments typical in ATFM systems.

4. Conclusions

In this study, we present a novel neural network approach to solve large-scale demand and capacity balancing within a collaborative ATFM framework. This method, based on deep learning, combines neural diving and neural branching to enhance solution quality. Neural diving is primarily used to find high-quality joint variable assignments, while neural branching narrows the gap between the best assignment and the target value of the optimal solution. Within our neural network method, traditional problems are treated as mixed-integer programming (MIP) problems, with solvers used to find approximate optimal solutions. We tested our method in a real-world complex operational scenario, focusing on Chinese national airspace. Compared to previous experiments, our approach uses a larger flight scale to identify the best GDP strategy. In the same collaborative ATFM framework, our method outperforms state-of-the-art RL-based and traditional DCB methods. Using 15,927 flight trajectories from a typical day in February 2024, spread across 287 airspace sectors with nominal capacity, we reallocated the entry and exit times of trajectories within each sector. The results show that large-scale ATFM problems can be optimally solved in just 15 minutes.

While this work demonstrates the potential of deep learning to solve real-world ATFM problems, it has limitations, notably the exclusion of airborne delays, uncertainty factors, sector opening schemes, and fairness considerations. Future research will explore how to enhance the model to include airborne delays, enabling a more comprehensive solution for air traffic flow management that accounts for delays across both ground and airborne phases of flight. Additionally, we plan to explore the use of cloud or edge computing to further improve scalability and performance, as well as predictive analytics and machine learning techniques to facilitate proactive adjustments to flight schedules.

References

1. Chen, Y.; Zhao, Y.; Wu, Y. Recent progress in air traffic flow management: A review. *Journal of Air Transport Management* **2024**, 116, 102573.
2. de l’aviation civile internationale, *O. Global Air Navigation Plan for CNS/ATM Systems*; ICAO, **2002**.
3. de l’aviation civile internationale, *O. Global air traffic management operational concept*; ICAO, **2005**.
4. de l’aviation civile internationale, O*. The Aviation System Block Upgrades, The Framework for global harmonization;* ICAO, **2016**.
5. Eurocontrol-SESAR, J. European ATM Master Plan-The Roadmap for Sustainable Air Traffic Management, **2012**.
6. Degas, A.; Islam, M.R.; Hurter, C.; Barua, S.; Rahman, H.; Poudel, M.; Ruscio, D.; Ahmed, M.U.; Begum, S.; Rahman, M.A.; et al. A survey on artificial intelligence (ai) and explainable ai in air traffic management: Current trends and development with future research trajectory. *Applied Sciences* **2022**, 12, 1295.
7. Yang, H.; Alphones, A.; Xiong, Z.; Niyato, D.; Zhao, J.; Wu, K. Artificial-intelligence-enabled intelligent 6G networks. *IEEE network* **2020**, 34, 272–280.
8. IBM ILOG CPLEX Optimization Studio, Version 12.10.0, IBM Corp*. https://www.ibm.com/docs/en/icos/12.10.0.* **2024**.11.
9. Gurobi Optimizer. Gurobi 12.0 Documentation. [*https://docs.gurobi.com/current/*](https://docs.gurobi.com/current/)*.* **2024**.11.
10. Forrest, John, and Robin Lougee-Heimer. "CBC user guide." *In Emerging theory, methods, and applications*, pp. 257-277. INFORMS, **2005**.
11. Holmström, K., Göran, A.O. and Edvall, M.M. USER’S GUIDE FOR TOMLAB/Xpress R2004. **2007.**
12. Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O’Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349* **2020**.
13. Xu, Y.; Dalmau, R.; Melgosa, M.; Montlaur, A.; Prats, X. A framework for collaborative air traffic flow management minimizing costs for airspace users: Enabling trajectory options and flexible pre-tactical delay management. *Transportation Research Part B: Methodological* **2020**, 134, 229–255.
14. Xu, Y.; Prats, X.; Delahaye, D. Synchronised demand-capacity balancing in collaborative air traffic flow management. *Transportation Research Part C: Emerging Technologies* **2020**, 114, 359–376.
15. Tang, Y.; Xu, Y. Multi-agent deep reinforcement learning for solving large-scale air traffic flow management problem: A time-step sequential decision approach. *In Proceedings of the 2021 IEEE/AIAA 40th digital avionics systems conference (DASC)*. IEEE, **2021**, pp. 1–10.
16. Chen, Y.; Xu, Y.; Hu, M.; Yang, L. Demand and capacity balancing technology based on multi-agent reinforcement learning. *In Proceedings of the 2021 IEEE/AIAA 40th digital avionics systems conference (DASC)*. IEEE, **2021**, pp. 1–9.
17. Yutong, C.; Minghua, H.; Yan, X.; Lei, Y. Locally generalised multi-agent reinforcement learning for demand and capacity balancing with customised neural networks. *Chinese Journal of Aeronautics* **2023**, 36, 338–353.
18. Crespo, A.M.F.; Weigang, L.; de Barros, A.G. Reinforcement learning agents to tactical air traffic flow management. *International Journal of Aviation Management* **2012**, 1, 145–161.
19. Kravaris, T.; Spatharis, C.; Blekas, K.; Vouros, G.A.; Garcia, J.M.C. Multiagent reinforcement learning methods for resolving demand-capacity imbalances. *In Proceedings of the 2018 IEEE/AIAA 37th digital avionics systems conference (DASC)*. IEEE, **2018**, pp. 1–10.
20. Spatharis, C.; Blekas, K.; Bastas, A.; Kravaris, T.; Vouros, G.A. Collaborative multiagent reinforcement learning schemes for air traffic management. *In Proceedings of the 2019 10th international conference on information, intelligence, systems and applications (IISA)*. IEEE, **2019**, pp. 1–8.
21. Spatharis, C.; Bastas, A.; Kravaris, T.; Blekas, K.; Vouros, G.A.; Cordero, J.M. Hierarchical multiagent reinforcement learning schemes for air traffic management. *Neural Computing and Applications* **2023**, pp. 1–13.
22. Mas-Pujol, S.; Salamí, E.; Pastor, E. Image-based multi-agent reinforcement learning for demand–capacity balancing. *Aerospace* **2022**, 9, 599.
23. Agogino, A.K.; Tumer, K. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems* **2012**, 24, 1–25.
24. Kravaris, T.; Vouros, G.A.; Spatharis, C.; Blekas, K.; Chalkiadakis, G.; Garcia, J.M.C. Learning policies for resolving demand- capacity imbalances during pre-tactical air traffic management. *In Proceedings of the Multiagent System Technologies: 15th German Conference*, MATES 2017, Leipzig, Germany, August 23–26, **2017**, Proceedings 15. Springer, 2017, pp. 238–255.
25. Spatharis, C.; Kravaris, T.; Vouros, G.A.; Blekas, K.; Chalkiadakis, G.; Garcia, J.M.C.; Fernandez, E.C. Multiagent reinforcement learning methods to resolve demand capacity balance problems. *In Proceedings of the Proceedings of the 10th hellenic conference on artificial intelligence*, **2018**, pp. 1–9.
26. Duong, T.; Todi, K.K.; Chaudhary, U.; Truong, H.L. Decentralizing air traffic flow management with blockchain-based reinforcement learning. *In Proceedings of the 2019 IEEE 17th international conference on Industrial informatics (INDIN)*. Ieee, **2019**, Vol. 1, pp. 1795–1800.
27. Huang, C.; Xu, Y. Integrated frameworks of unsupervised, supervised and reinforcement learning for solving air traffic flow management problem. *In Proceedings of the 2021 IEEE/AIAA 40th digital avionics systems conference (DASC)*. IEEE, **2021**, pp. 1–10.
28. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research* **2021**, 290, 405–421.
29. Achterberg, Tobias. SCIP: solving constraint integer programs. *Mathematical Programming Computation* **2009**, 1, 1–41.
30. Khalil, Elias and Le Bodic, Pierre and Song, Le and Nemhauser, George and Dilkina, Bistra. Learning to branch in mixed integer programming. *Proceedings of the AAAI conference on artificial intelligence* **2016**, 30, 1.
31. Fischetti, Matteo and Jo, Jason. Deep neural networks and mixed integer linear optimization. *Constraints* **2018**, 23, 296–309.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

1. www.Variflight.com [↑](#footnote-ref-1)