



Flight Arrival Scheduling via Large Language Model

Wentao Zhou¹, Jinlin Wang², Longtao Zhu² , Yi Wang² and Yulong Ji^{1,*} 

¹ School of Aeronautics and Astronautics, Sichuan University, Yihuan Road, Chengdu 610065, China; zhouwentao@stu.scu.edu.cn

² College of Computer Science, Sichuan University, Yihuan Road, Chengdu 610065, China; jinlinwang@stu.scu.edu.cn (J.W.); zhulongtao@stu.scu.edu.cn (L.Z.); wangyi_aa@stu.scu.edu.cn (Y.W.)

* Correspondence: jy1@scu.edu.cn

Abstract: The flight arrival scheduling problem is one of the critical tasks in air traffic operations, aiming to ensure that the flight arrive in the correct sequence safely. Existing methods primarily focus on the terminal area and often overlook the presence of training flight at the airport. Due to the limited generalization of traditional methods and varying control practices at different airports, training flight at airports still rely on manual control for arrival sorting. To effectively address these issues, we propose a novel method for slot allocation that leverages the strong reasoning capabilities and generalization potential of large language models (LLMs). Our method conceptualizes the dynamic scheduling problem for training flight as a language modeling problem, a perspective not previously explored. Specifically, we represent the allocator's inputs and outputs as language tokens, utilizing LLMs to generate conflict-free results based on a language description of requested landing information and assigned training flight information. Additionally, we employ a reset strategy to create a small dataset for scenario-specific samples, enabling LLMs to quickly learn allocation schemes from the dataset. We demonstrated the capability of LLMs in addressing time conflicts by evaluating metrics such as answer accuracy, conflict rate, and total delay time (without the wrong answer). These findings underscore the feasibility of employing LLMs in the field of air traffic control.

Keywords: LLMs; supervised fine-tuning; arrival scheduling; language modeling



Citation: Zhou, W.; Wang, J.; Zhu, L.; Wang, Y.; Ji, Y. Flight Arrival Scheduling via Large Language Model. *Aerospace* **2024**, *11*, 813. <https://doi.org/10.3390/aerospace11100813>

Academic Editor: Michael Schultz

Received: 25 August 2024

Revised: 22 September 2024

Accepted: 2 October 2024

Published: 6 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The flight arrival scheduling problem aims to allocate landing slots to flight based on preset rules and determine their actual arrival times, ensuring both the safety of the flight and the efficiency of airport operations. Different runway occupancy times and safe separation distance for each flight type (i.e., light, medium, and heavy plane) make it challenging to assign arrival sequences effectively. An unreasonable assignment can negatively impact airport resource utilization, increase flight delays, and even lead to flight plan cancellation. Therefore, it is crucial to develop a dynamic scheduling method that could reduce flight delays and ensure the orderly landing of flight [1,2]. This challenge has garnered significant attention from scholars both domestically and internationally.

Currently, due to the influence of wake vortices and route crossings, most small and medium-sized airports utilize only a single runway for landing at the same time [3,4]. The efficient utilization of runway resources is critical for the implementation of flight plans. Conflicts between standard terminal arrival routes (STARs) [5] greatly impact the safety and efficiency of training flight's landing when departing from the airspace. Furthermore, the slow development of air traffic control systems and regional differences in small and medium-sized airports have hindered the implementation of Airport Collaborative Decision Making (A-CDM) and flight schedule monitoring [6]. Consequently, most small and medium-sized flights still rely on manual scheduling, which is time-consuming and prone to errors.

The flight arrival scheduling problem is classified as NP-hard [7], meaning that the complexity of finding an optimal solution grows exponentially with the problem size. This inherent complexity poses significant challenges for real-time or near-real-time optimization, as computational resources and time constraints limit the feasibility of exact solutions. As a result, practical approaches often focus on identifying near-optimal solutions within reasonable time frames.

Among these approaches, reinforcement learning (RL) has gained attention as a promising method due to its ability to learn from dynamic environments and adapt scheduling decisions based on feedback. However, despite its potential, RL still faces significant challenges when applied to flight scheduling. These challenges include fixed action, the requirement for extensive training time, slow convergence rates, and difficulties in generating high-quality solutions. These limitations hinder the practical application of RL in scenarios demanding reliable scheduling adjustments.

To address these challenges, this article proposes a novel approach to flight arrival scheduling utilizing supervised fine-tuning of LLMs. This approach enables the LLM to learn arrival scheduling rules from a given dataset, requiring minimal data and time for completion. The main contributions of this article are as follows:

1. By using a reset strategy [8,9] to simulate the reinforcement learning training process, a fine-tuning dataset is produced. This method effectively simulates rule-based slot conflict scenarios, reducing the production costs and fine-tuning time;
2. This article represents the first instance in the industry of applying LLMs to solve the arrival scheduling problem by converting it into a language modeling problem. By leveraging historical scheduling data, the LLM can learn control habits, thus paving a new path for the application of LLMs in this industry;

The rest of the paper is organized as follows: In Section 2, we review the related work, including arrival slot allocators and the application of LLMs in arrival sequencing. In Section 3, we outline the methodology, covering problem definitions, the slot allocator as a language model, supervised fine-tuning, and the conflict resolution process. Section 4 details the experiments, including dataset generation and assessment methods. Finally, Section 5 concludes the paper and suggests directions for future work.

2. Literature Review

2.1. Arrival Slot Allocator

The problem of flight arrival sequencing has existed for a long time. Many scholars have been studying the scheduling problem of arrival sequencing conflicts [10–12]. Two commonly used methods to address the problem of arrival sequencing conflicts are FCFS and priority-based dynamic scheduling. FCFS is a static scheduling method that prioritizes flight based on their order of arrival. Due to its simplicity and early adoption in flight scheduling, FCFS serves as a baseline control group in flight arrival experiments. Priority-based scheduling dynamically, on the other hand, dynamically adjusts the arrival order according to the priority of specific scheduling rules [13]. This is a constrained dynamic scheduling process, such as shortest remaining time first (SRTF) [14], fairness priority, etc., solved by various algorithms, including genetic algorithms [15–17], particle swarm algorithms [18–20], fish swarm algorithms [21,22], and reinforcement learning algorithm [23,24]. For instance, Phillips et al. used linear programming relaxations to minimize the mean arrival time [25], while Ming Wei et al. proposed a multi-objective mixed integer linear programming model to maximize flight delays under different runway usage patterns [26]. Hu X B designed a binary-based genetic algorithm to minimizing the total delay time [27]. Androutopoulos et al. aims for delay fairness within a limited displacement range and optimized flight delay displacement size [28]. Kaiquan Cai et al. utilized the multi-agent reinforcement learning algorithms to explore the connection between delay fairness and total delay time [29], though this method requires substantial training time and data.

In this study, we leverage the generalization and inference capabilities of LLMs to address the scenario of unordered arrivals from different airspaces with a relatively small

dataset, This approach not only assists in slot arrangement but also enables the LLM to learn conflict-free strategies by introducing a reasoning mechanism as part of the fine-tuning process. Additionally, fine-tuning requires only a short period of time and exhibits good stability and efficiency.

2.2. LLM on Arrival Sequence

LLMs such as ERNIE [30], LLaMa [31], etc. were capable of understanding and generating human language, performing tasks such as text summarization, question answering, translation, and generative writing, which presents potential and challenges for various industries. More and more scholars are using LLMs for relevant research, such as Jiageng Mao et al. ([32]) using LLMs in the field of autonomous driving to enhance the generalization and explainability of autonomous driving in new scenarios. Thirunavukarasu ([33]) et al. utilized LLMs to explore their application in healthcare settings, aiming to improve efficiency and effectiveness in clinical, educational, and research work in medicine. Zhang, Q, et al. ([34]) used LLMs to reconstruct flight trajectories based on ADS-B data, which opens up numerous opportunities for LLMs in the field of air transportation. Abdulhak S ([35]) utilized LLMs to optimize Ground Delay Programs (GDPs) by training CHATATC on extensive historical GDP data from 2000–2023, showcasing the transformative potential of LLMs in refining strategic traffic flow management. Jimmy Thatcher ([36]) utilizes artificial intelligence, LLMs, and advanced analytics to reduce the emission footprints and costs linked to helicopters for regional oil operations, while enhancing productivity, safety, and efficiency. Wang ([37]) used AviationGPT, built on open-source LLaMA-2 and Mistral architectures and trained on curated aviation datasets, to address the problem of effectively utilizing complex, unstructured text data in the aviation industry. This solution improves Natural Language Processing (NLP) performance, enabling better handling of diverse tasks and enhancing the efficiency and safety of National Airspace System (NAS) operations.

In this paper, the authors address the arrival sequence problem by reformulating it as a language modeling problem and solve the allocation problem of the arrival slot by fine-tuning, focusing on dynamic scheduling and being able to adjust the allocation of the approach slot in real-time. In contrast to the advanced actions set by reinforcement learning to specify the delay time of the flight, LLMs can learn the dynamic deployment of the flight queue entirely. The results of arrival slot scheduling are represented as multiple sets of numerical results for approach time slot allocation. This poses a significant challenge for the large language models, as they need to effectively demonstrate their numerical reasoning and planning capabilities in air traffic control.

3. Methodology

In this subsection of this paper, the authors define the problem of arrival scheduling, formulate the arrival scheduling issue as a language modeling process, discuss the application of supervised fine-tuning (SFT), and introduce the process of conflict resolution.

3.1. Problems Definition

To facilitate model presentation, all definitions and notations used hereafter are summarized in Table 1.

Arrival scheduling is to allow flights to take off in an orderly manner according to safe time intervals, which could be represented by using the time slot. It is necessary to ensure the order and low delay of training flight and the high efficiency of airport operation with different flight's types, different wake turbulence vortex, and different airspace. The constraints are as follows:

- The time for the flight to complete the training is known;
- The flight flies at a constant speed;

Table 1. Parameters and variables in the proposed model.

Indices:	
i, j	Index for a flight
t	Index for time slot
Sets:	
F	Set of all flights that have completed the training task
T_i	Set of time slots for flight i , $\{T_{Start}, T_{End}\}$
$Slot_{info}$	Set of information related to the flight's training completion
$Slot_{History}$	Set of historical time slot information provided to the large language model
$Slot_{Results}$	Set of actual time slot allocations by LLMs
Parameters:	
t_i^r	Slot requested by flight i
t_i^a	Actual slot allocated to flight i
t_i^T	Scheduled time for flight i considering wake vortex constraints
C_T	Safe time interval constant between different flight types
K_T	Safe wake vortex time interval between two adjacent flights
T_{Start}	Start time of the airport operating period
T_{End}	End time of the airport operating period
F_{ID}	ID of the flight
F_{type}	Type of the flight
$F_{request_slot}$	Application for the time slot when the flight completes the training
F_{real_slot}	Real allocated time slot for the flight
F_{delay}	Delay time, difference between the requested and actual allocated time
$Epdone$	Flag to clear historical time slot information and key to data slicing for LLMs
Decision Variables:	
S_t	Decision variable indicating whether time slot t is occupied
F_i	Information set for each flight in the results, including ID, type, requested slot, real slot, delay, and $Epdone$

The objective function for arrival scheduling is as follows:

$$\min Z = \sum_{i \in F} t_i^a - t_i^r \quad (1)$$

The constraints are as follows:

$$\sum_{i, j \in F} t_i^a - t_j^a > C_T \quad (2)$$

$$\sum_{i, j \in F} t_i^T - t_j^T > K_T \quad (3)$$

$$\begin{cases} \sum_{t \in T_i} S_t = 1 \\ \sum_{i \in F} t_i^a - t_i^r \geq 0 \\ T_i = \{T_{Start}, T_{End}\}, i = 1, 2, \dots, m \end{cases} \quad (4)$$

$$S_t = \begin{cases} 1, & \text{if slot } t \text{ is assigned} \\ 0, & \text{if slot } t \text{ is free} \end{cases} \quad (5)$$

Equation (1) is the minimum total delay time for the training flight. Equation (2) is the safe time interval constraint between two adjacent flights. Equation (3) is the safe time

interval of the wake vortex constraint between two adjacent flights. Equation (4) is the validity constraint of time slot arrangement.

The time slots utilized for the arrival sequence are illustrated in Figure 1, depicting the flight returning from the airspace and arriving in accordance with the standard arrival procedure. Equidistant time nodes are set along the return route with intervals of 1 min, and the flight completing the task would follow the landing guidance according to the assigned slot, as depicted by the two blue flight in the image. Different types of flight need to undergo safety inspections at assigned time slot nodes based on flight time interval constraints. For example, green flights and red flights return from their respective airspaces at the same time. Airspace A only takes four minutes to reach the waypoint intersection, while airspace B takes five minutes. If the safe time interval required by the green flight in airspace A exceeds 1 min, the two flights will conflict at this time and result in delays.

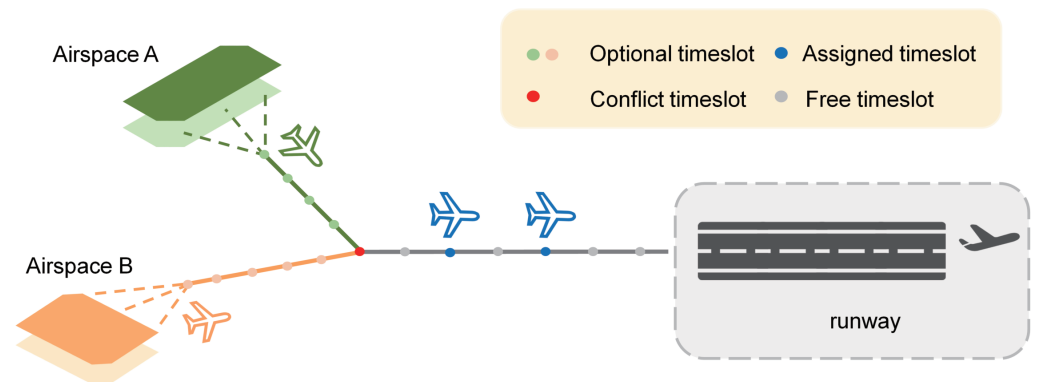


Figure 1. Safety Slot Interval Diagram. The safe time interval division forms the foundation of arrival sorting. This figure illustrates the division of slot points along the planned route from the airspace exit point to the airport. These slot points are categorized into: optional timeslot, which require application based on occupancy status; assigned timeslot, already taken by other flights; conflict timeslot, which do not meet the safe time interval criteria relative to assigned timeslot; and unoccupied, free timeslot.

3.2. Slot Allocator as Language Modeling

The key to this section is how to reformulate the arrival scheduling problem as a language modeling problem. Features are extracted from the training flight's information to construct the language model for the time slot problem as shown in Figure 2. The feature vector of the model inputs is represented as follows:

$$Input = \{Slot_{info}, Slot_{History}\} \quad (6)$$

$$Slot_{info} = [F_{ID}, F_{type}, F_{request_slot}, E_{pdone}] \quad (7)$$

The provided contextual text for the large language model $Slot_{History}$ is as follows:

$$Slot_{History} = [F_{ID}, F_{type}, F_{request_slot}, E_{pdone}] \quad (8)$$

The feature vectors of the large language mode output are represented as follows:

$$Slot_{Results} = [F_1, F_2, F_3, \dots, F_n, F_{n+1}] \quad (9)$$

$$F_i = [F_{ID}, F_{type}, F_{request_slot}, F_{real_slot}, F_{delay}, E_{pdone}], F_i \in Slot_{Results} \quad (10)$$

Equation (6) represents the input feature vector of the large language model. Equation (9) represents the output feature vector of the large language model. Equation (8) will be used as the input feature vector of the large language model in next epoch.

```

**Airport Flight Slot Allocator**
Role: You are the coordinator for managing airport flight slots. Allocate slots to flights based on their schedules and requests.

Context
- Airport Layout: Manage airport single runway takeoff and landing time slot occupancy.
- Objective: Efficiently allocate slots to maximize airport throughput and minimize delays.

Rules
- Listed from front to back according to rule priority
1. The safe time interval is one minute for flight type J, two minutes for flight type Y, and 3 minutes for flight type B.
2. In the case of time slot occupancy conflict, the priority of J is greater than that of Y and that of B.
3. The priority of local landing flights is higher than that of local flights.

Inputs
1. Flight Requests: Information about flights needing slots, includes the flight ID, flight model, origin, requested slot, takeon/off,
episode done.
2. Historical Data: Past slot allocations and their outcomes for false of episode done.
3. Operational Priorities: Specific goals or priorities, such as minimizing delays or prioritizing certain flight types.

Task
- Analysis: Assess the feasibility of requests based on current constraints and priorities.
- Decision Making: Allocate slots or suggest modifications to flight schedules.

Output
- Slot Allocation:
  - Detailed slot times and information for each flight, a tuple includes requested_slot, real_slot, flight_id, delay_time, ep_done
  """

```

Figure 2. Prompt of Language Modeling. The prompt outlines the assistant’s message, tasks, rules, and the inputs and outputs that help achieve its objectives.

3.3. Supervised Fine-Tuning

SFT (Supervised Fine-Tuning) is a technique used in deep learning, particularly in the field of refining large pre-trained models, that aims to adapt pre-trained models to specific tasks or datasets by adjusting model parameters guided by labeled data. By leveraging pre-trained model knowledge and features, SFT reduces training costs and time compared to training models from scratch.

In this paper, the authors select three LLMs as mutual controls, respectively, the ERNIE Lite, LLAMA-2, and BLOOMZ, and the mixed linear integer programming (MILP) training data are utilized as the sample dataset for the fine-tuning. The fine-tuning parameters are shown in Table 2.

Table 2. Fine-tuning parameters table.

Training Method	Epoch	Learning Rate	Data Splitting
Full Refresh	3	0.00003	10%

Training method: The parameters of the large model fine-tuning are set for full updates, offering greater adaptability to specific task training compared to LoRA (Large Offsite Recloning Activation). LoRA, in contrast, concentrates on updating only a subset of model parameters while maintaining the rest unchanged, leading to improved efficiency and fine-tuning feasibility, particularly in resource-constrained environments.

Epoch: Epochs refer to the number of times the entire dataset is passed forward and backward through the neural network during training, which is selected based on the size of the training data and changes in the loss function.

Learning Rate: The learning rate determines the step size at which the model parameters are updated during training. A smaller learning rate like 0.00003 indicates slower and more cautious updates, which can be beneficial for fine-tuning large pre-trained models without causing drastic changes that may disrupt their learned representations.

Data splitting: By splitting 10% of the fine-tuning dataset as a validation dataset, which serves as an objective baseline in the fine-tuning process, the model’s performance, such as accuracy, precision, and recall, is quantified by comparing the model’s predicted results with the expected results on other evaluation data. The evaluation dataset not only assists in assessing the model’s predictive and generalization capabilities but also helps to prevent overfitting.

3.4. Conflict Resolution Process

The conflict resolution process includes conflict detection, conflict adjustment, and safe time interval judgment.

Conflict detection: Conflict detection is the initial phase in the conflict resolution process, mainly based on the initial scheduling request information and historical information. The goal is to identify situations where multiple flight request the same time period. For example, if flight A and B apply for the same slot simultaneously, it will cause a scheduling conflict. During this phase, all conflict situations are identified and logged.

Conflict Adjustment Once a conflict is detected, conflict adjustment follows. Adjustments are handled based on the type of flight and its priority. Flights with higher priority remain on the current schedule, while flights with lower priority are adjusted to other time slots. The adjustment principle is to minimize delays. For instance, since the time gap required behind a heavy flight is usually longer than that behind a small flight, priority may be given to keeping the small flight in the original time slot while the heavy flight is adjusted.

Interval Judgment: The final phase is to judge the safe time interval. The goal is to ensure that the adjusted scheduling plan meets the preset safe time interval. At this stage, the new sequencing results are evaluated to ensure that all flight arrival intervals are within safe limits.

As depicted in Figure 3, both flight A and B request the same time slot t_1 concurrently, resulting in a detected conflict. Prioritizing flight A to maintain its current schedule incurs less delay compared to accommodating flight B, as the time gap required behind a heavy flight is typically longer than that behind a light one, aligning with the preset safety time intervals. Following conflict adjustment, the revised arrival sequence is ACB and the time slot is $t_1t_2t_3$, with the time slot scheduled to $t_1t_2t_4$ accordingly to adhere to the preset safety time intervals.

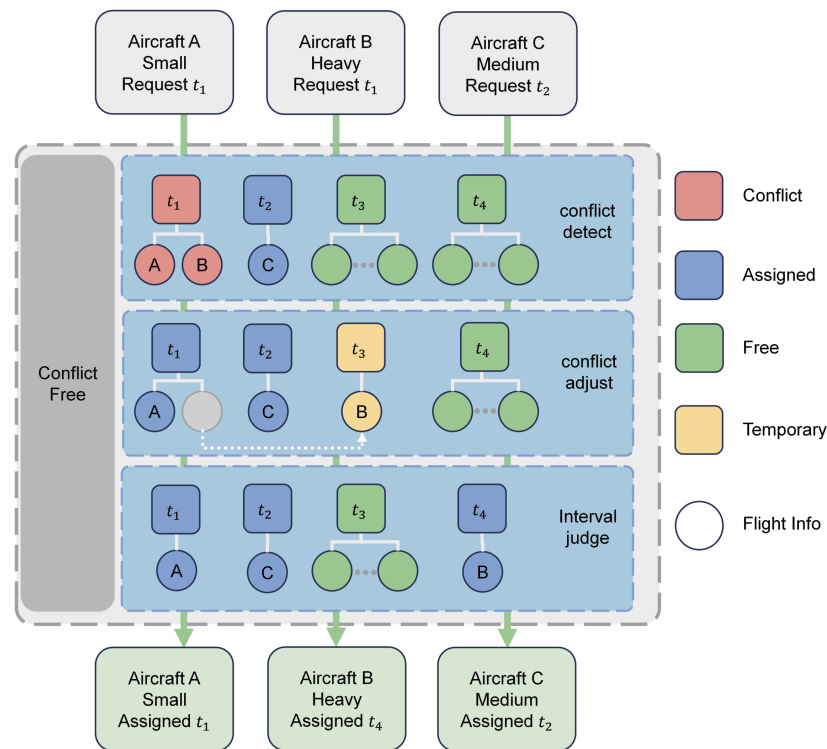


Figure 3. Slot Conflict Allocator Process Diagram. The process mainly shows the algorithmic part of the fine-tuning dataset and validation dataset production, including conflict detection, conflict adjustment, and safe time interval judgement, and the process of data change.

4. Experiment

4.1. Dataset Generation

The simulation created an airport and several surrounding hypothetical airspaces. For example, the training flight flies into airspace according to a custom plan. When it completes the task and returns from the airspace, the simulation software BlueSky-simulator==2022.12.22 calculates the time needed for the training flight to return from its current position to the merge point in different airspaces. This time is regarded as the requested time of the training flight, and the return information is sent to the sorting decision-making end in the language modeling input format to obtain a conflict-free solution, as shown in Figures 4 and 5.

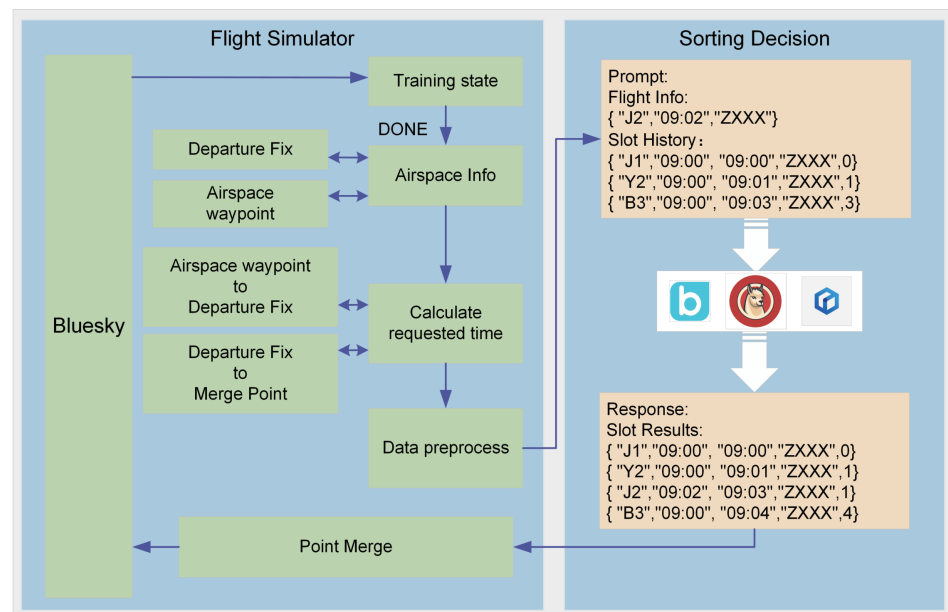


Figure 4. Sorting Verification Flow Chart. The sequencing decision-making process primarily involves replacing the arrival sequencing time decision-making component of the flight simulation with decision-making driven by LLMs. The aircraft return application information is processed and transmitted to the LLM, which then provides decision-making results. This information is extracted from the text by the flight simulation system, and based on the decision, delays are assigned to the pre-set waypoints accordingly.

To ensure the generalization of the LLM allocator so that it can quickly learn the deployment rules and efficiently deploy the total flight plan, this study employs a reset strategy to conduct conflict simulations for different flight types and time slot allocation scenarios. These scenarios mainly target three categories: cross-time scenarios, safe interval scenarios, and minimize delay scenarios.

Cross-time scenarios: A significant consideration is the handling of cross-time scenarios. For instance, if a conflict occurs at 8:59, the LLM might erroneously schedule the flight to 8:62. Since this timing is not valid, it is crucial to perform targeted simulations for flight time slots that span across time boundaries to ensure realistic and feasible scheduling.

Safe interval scenarios: In the realm of arrival scheduling, adherence to safety time intervals is paramount to ensuring orderly and conflict-free sequences. To facilitate clear comprehension by the LLM, predefined safe time interval constraints must be established for various types of flight. For instance, between small and medium-sized flight, if a small flight is assigned the 9 o'clock time slot, subsequent flight must maintain a minimum safe time interval of 1 min after the small flight, and a minimum safe time interval of 2 min after medium-sized flight, and so forth.

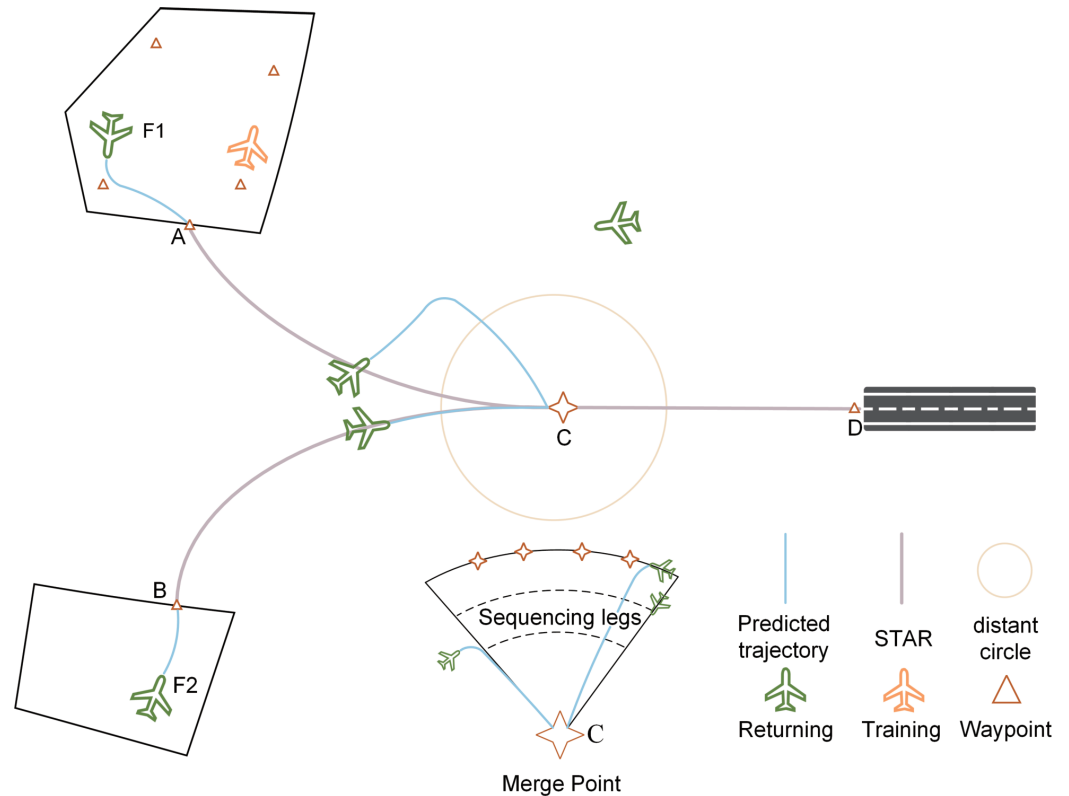


Figure 5. Flight Simulation Schematic. This schematic diagram represents the entire flight simulation, using two colors to indicate the aircraft's state. Upon completing the flight training mission, the aircraft returns by following preset segments. The merge point of the return path set the different waypoints at the various distances circle is to absorb delay times.

Minimize delay scenarios: According to the established mathematical model, the resequencing of training flight for arrival not only needs to consider safe time interval constraints but also aims to minimize delay. Flight simulations are required to simulate scenarios where different flight types apply to the same slot. For example, when both small flight A and medium-sized flight B request to occupy the same time slot, the delay caused by A assigning the slot versus B assigning it is not equivalent. As shown in Figure 6, LLMs are capable of determining whether the existing time slot allocation needs adjustment based on the prompt information and providing the adjusted result. If the dataset is made from FCFS, just like in Figure 6b, the total delay is higher than the MILP like in Figure 6a.

The number of arriving flight at the airport each day exceeds 30, but the slot conflict scenario encountered by these 30 flights may not be ideal. To minimize the time and economic costs associated with fine-tuning, this study did not set the reward function. Instead of the exploration strategy, a reset strategy pair was used. By utilizing the 'epdone' parameter of the reset strategy to set specific conflict scenarios, the conflict situations faced in the flight plan are modeled more precisely.

The representative conflict data is filtered and datasets are created based on the designed scenarios. The historical data gradually accumulates until it is cleared when the *Epdone* parameter is true, as shown in the Figure 7 below.

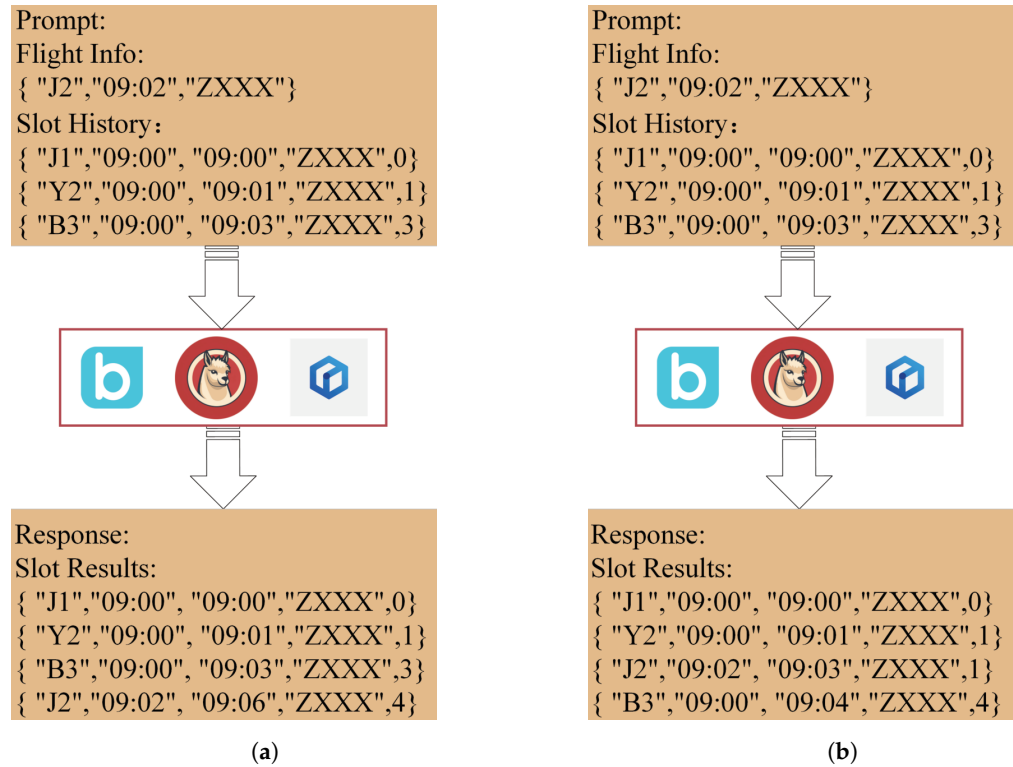


Figure 6. Minimize delay scenarios. Different scenarios result in different delay outcomes, with results derived using FCFS as the fine-tuned dataset in (a) and MILP as the fine-tuned dataset in (b).

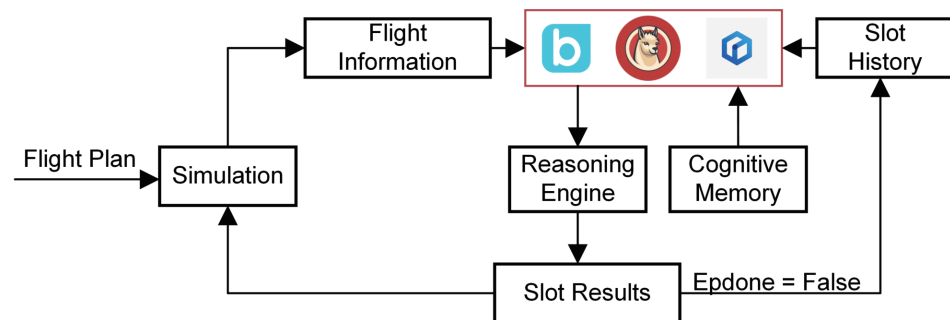


Figure 7. Reset strategy Diagram. During the fine-tuning process, a reinforcement learning training strategy is used to extract flight arrival conflict scenarios, and during the training process, the LLM stores assigned time-slot information based on predetermined end-of-round markers, and all training enhances the cognitive memory and reasoning ability of the large model gradually.

4.2. Assessment

The preliminary evaluation of the LLMs was carried out after the end of SFT, and the results are shown in Table 3.

Table 3. Preliminary evaluation of MILP.

Model	Accuracy	F1 Score	BLEU-4
ERNIE-Lite	61.04%	99.82%	99.55%
LLaMa-2	57.14%	99.31%	98.51%
BLOOMZ	48.05%	99.01%	97.88%

Accuracy in Table 3 indicates the degree to which the answers from fine-tuned LLMs are the same as the answers from MILP with the same input consisting of flight information and historical information. The F1 score is the reconciled average of the precision and

recall in the answers given by the large language model, ignoring the stop words. BLEU-4 is commonly used for the generative class of tasks such as machine translation/text summarization in NLP evaluation metrics, which represent the weighted average precision rate between the generated results and the labeled results.

It is evident that the accuracy, F1 score, and BLEU-4 score of ERNIE-Lite after SFT exceed those of the other two large models, standing at 61.04% accuracy, 99.82% F1 score, and 99.55% BLEU-4 score. To further test the ability of the large language model to accomplish the arrival sequencing task after SFT, the authors use the eval dataset, which is the same as that of the preliminary evaluation, to compare the differences between LLMs and MILP alongside FCFS methods. Among the results of 78 sets of conversations with LLMs, the comparison results between LLMs and MILP are illustrated by the conflict rate, the correctness rate, and the total delay without wrong conversation (TDW), as shown in Table 4 below.

Table 4. Comparison of methods.

Method	Conflict Rate	Correctness Rate	TDW		
			LLM	FCFS	MILP
ERNIE-Lite	35.3%	64.7%	478	230	188
LLAMA-2	23.17%	76.83%	423	453	372
BLOOMZ	21.95%	78.05%	449	466	382

In Table 4, it is observed that the BLOOMZ, after being fine-tuned, exhibits a sorting conflict rate of 21.95% and an answer accuracy rate of 78.05%. The learning effect is better than LLaMa-2 and ERNIE-Lite, despite the preliminary accuracy rate being only 48.05%. The difference mainly comes from some of these data being inconsistent with the expected results, such as the sort orders AB and BA, which adhered to the safe interval rule and were also judged to be wrong in the first evaluation.

The performance of ERNIE-Lite falls short compared to LLaMa-2 and BLOOMZ in terms of processing ability and learning capability for complex arrival sequencing problems, as indicated by the total delay time (excluding conflicting conversation results)/FCFS/MILP indicators, which reflect the complexity of the conflict.

This discrepancy arises from the inconsistency in the order of questions answered incorrectly by LLMs. For instance, among 78 dialogues, ERNIE might misinterpret the 1st, 13th, and 17th dialogues, while BLOOMZ might err on the 2nd, 27th, and 68th dialogues. Such variations stem from the differing complexity of conversation data concerning time slot conflicts, thus impacting the TDW differently for each model.

Consequently, the authors filtered out incorrectly answered questions by the LLMs and then calculated the TDW values of various methods. TDW evaluates the LLMs' learning ability for conflict resolution via the numerical ratio of LLM/FCFS/MILP. Higher TDW values of FCFS and MILP indicate more intricate scenarios for correct conflict resolution by the LLMs. If the TDW numerical values produced by LLMs are closer to those obtained by the MILP method for the dataset, it indicates that the learning performance of LLMs on the dataset is better.

5. Conclusions and Future Work

In this paper, we proposed a method using SFT to address time slot conflicts, demonstrating the feasibility of employing LLMs as a tool for resolving scheduling challenges in air traffic control. The results suggest that LLMs can provide a more flexible, data-driven solution for flight scheduling. However, the current performance of LLMs remains similar to basic methods such as FCFS, indicating substantial potential for further enhancement.

LLMs distinguish themselves from rule-based and data-driven methods by leveraging natural language processing to construct knowledge models. This unique capability enables LLMs to manage intricate scheduling tasks more effectively, positioning their use in arrival

sequencing as a significant step toward broader applications in air traffic control and other complex fields. Nevertheless, LLMs also have notable limitations, including high computational requirements, slower real-time response compared to traditional algorithms, and the risk of generating inaccurate or misleading outputs (hallucinations). These challenges highlight the need for ongoing research to improve the real-time performance and reliability of LLMs, ensuring they can meet the demands of dynamic, high-stakes environments.

Future research could focus on further exploring and developing the potential of fine-tuned LLMs in air traffic control, particularly in the following areas:

1. How to improve the LLMs' ability to sort the large number of flights when facing the time conflict problem? Through experiments, we can see that the ability of the large model to solve the problem of arrival deployment is only close to FCFS, and there is huge room for improvement.
2. Complex arrival scheduling conditions need to take into account flight delay prediction [38], airline crew scheduling [39], fuel consumption [40], flight maintenance [41], aircraft performance, and others. How to leverage LLMs to act as an arrival scheduler in complex situations? This is the main direction of future research.

Author Contributions: Conceptualization, W.Z. and J.W.; Methodology, W.Z., L.Z., Y.W. and Y.J.; Software, Y.W.; Data Curation, W.Z. and L.Z.; Writing—Original Draft Preparation, W.Z.; Writing—Review & Editing, W.Z., J.W., Y.W. and Y.J.; Supervision, L.Z. and Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used in this study is publicly available on GitHub at <https://github.com/wayne7412/Flight-Arrival-Schedule> (accessed on 21 September 2024). The current fine-tuning dataset is at least twice the size of the dataset originally used in the experiments. Users are advised to divide the dataset into 5/6 for fine-tuning and 1/6 for testing as appropriate, or extract approximately 150 dialogues from the fine-tuning dataset to serve as a test set for evaluation purposes.

Conflicts of Interest: The authors declare no conflicts of interest

References

1. Bennell, J.A.; Mesgarpour, M.; Potts, C.N. Dynamic scheduling of flight landings. *Eur. J. Oper. Res.* **2017**, *258*, 315–327. [[CrossRef](#)]
2. Dear, R.G. *The Dynamic Scheduling of Flight in the Near Terminal Area*; Technical Report; Flight Transportation Laboratory, Institute of Technology: Cambridge, MA, USA, 1976. Available online: <http://hdl.handle.net/1721.1/67982> (accessed on 22 September 1976).
3. Lieder, A.; Stolletz, R. Scheduling flight take-offs and landings on interdependent and heterogeneous runways. *Transp. Res. Part E Logist. Transp. Rev.* **2016**, *88*, 167–188. [[CrossRef](#)]
4. Messaoud, M.B. A thorough review of flight landing operation from practical and theoretical standpoints at an airport which may include a single or multiple runways. *Appl. Soft Comput.* **2021**, *98*, 106853. [[CrossRef](#)]
5. Chandra, A.; Choubey, N.; Verma, A.; Sooraj, K.P. Quasi-stochastic optimization model for time-based arrival scheduling considering Standard Terminal Arrival (STAR) track time and a new delay-conflict relationship. *J. Air Transp. Manag.* **2024**, *115*, 102527. [[CrossRef](#)]
6. Fitrianti, R.; Malkhamah, S.; Djunaedi, A.; Dewanti, D. Develop model harmonization ATFM and A-CDM with integrated policy network supporting air traffic services. *Int. J. Soc. Serv. Res.* **2024**, *4*, 1179–1195. [[CrossRef](#)]
7. Liu, W.; Delahaye, D.; Cetek, F.A.; Zhao, Q.; Notry, P. Comparison of performance between PMS and trombone arrival route topologies in terminal maneuvering area. *J. Air Transp. Manag.* **2024**, *115*, 102532. [[CrossRef](#)]
8. Chang, J.D.; Zhan, W.; Oertell, O.; Brantley, K.; Misra, D.; Lee, J.D.; Sun, W. Dataset reset policy optimization for RLHF. *arXiv* **2024**, arXiv:2404.08495. [[CrossRef](#)]
9. Asadi, K.; Fakoor, R.; Sabach, S. Resetting the optimizer in deep RL: An empirical study. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 9–15 December 2023.
10. Gui, D.; Le, M.; Huang, Z.; Zhang, J.; D'Ariano, A. Optimal flight arrival scheduling with continuous descent operations in busy terminal maneuvering areas. *J. Air Transp. Manag.* **2023**, *107*, 102344. [[CrossRef](#)]

11. Liu, R.; Piplani, R.; Toro, C. A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem. *Comput. Oper. Res.* **2023**, *159*, 106294. [[CrossRef](#)]
12. Peng, W.; Yu, D.; Lin, J. Resource-Constrained Multi-Project Reactive Scheduling Problem with New Project Arrival. *IEEE Access* **2023**, *11*, 64370–64382. [[CrossRef](#)]
13. Ma, X.; He, Z.; Yang, P.; Liao, X.; Liu, W. Agent-based modelling and simulation for life-cycle airport flight planning and scheduling. *J. Simul.* **2024**, *18*, 15–28. [[CrossRef](#)]
14. Li, Z.; Hu, Y.; Tian, L.; Lv, Z. Packet rank-aware active queue management for programmable flow scheduling. *Comput. Netw.* **2023**, *225*, 109632. [[CrossRef](#)]
15. Derviş, S.; Demir, H.I. Airline passenger planes arrival and departure plan synchronization and optimization using genetic algorithms. In Proceedings of the International Symposium on Intelligent Manufacturing and Service Systems, Istanbul, Turkey, 26–28 May 2023; Springer Nature: Singapore, 2023; pp. 413–423. [[CrossRef](#)]
16. Deng, W.; Li, K.; Zhao, H. A flight arrival time prediction method based on cluster clustering-based modular with deep neural network. *IEEE Trans. Intell. Transp. Syst.* **2023**, *25*, 6238–6247. [[CrossRef](#)]
17. Chen, K.; Wu, J.; Yang, L. Reassigning departure slots with preferences of the airline and passengers. *Transp. Res. Rec.* **2024**, *2678*, 786–804. [[CrossRef](#)]
18. Ribeiro, V.F.; Pamplona, D.A.; Fregnani, J.A.T.G.; de Oliveira, I.R.; Li, W. Modeling the swarm optimization to build effective continuous descent arrival sequences. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 760–765. [[CrossRef](#)]
19. Girish, B.S. An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the flight landing problem. *Appl. Soft Comput.* **2016**, *44*, 200–221. [[CrossRef](#)]
20. Deng, W.; Zhang, L.; Zhou, X.; Zhou, Y.; Sun, Y.; Zhu, W.; Chen, H.; Deng, W.; Chen, H.; Zhao, H. Multi-strategy particle swarm and ant colony hybrid optimization for airport taxiway planning problem. *Inf. Sci.* **2022**, *612*, 576–593. [[CrossRef](#)]
21. Bing, D.; Wen, D. Scheduling arrival aircrafts on multi-runway based on an improved artificial fish swarm algorithm. In Proceedings of the 2010 International Conference on Computational and Information Sciences, Chengdu, China, 17–19 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 499–502. [[CrossRef](#)]
22. Li, J.; Liang, W.; Li, Y. Research on flight scheduling of two runways based on the fusion of artificial fish school algorithm and genetic algorithm. In Proceedings of the 2014 IEEE 5th International Conference on Software Engineering and Service Science, Beijing, China, 27–29 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 279–282. [[CrossRef](#)]
23. Ali, H.; Thinh, P.D.; Alam, S. Deep reinforcement learning based airport departure metering. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 366–371. [[CrossRef](#)]
24. Limin, S.; Due-Thinh, P.; Alam, S. Multi-agent deep reinforcement learning for mix-mode runway sequencing. In Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, 8–12 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 586–593.
25. Phillips, C.; Stein, C.; Wein, J. Scheduling jobs that arrive over time. In Proceedings of the Workshop on Algorithms and Data Structures, Kingston, ON, Canada, 16–18 August 1995; Springer: Berlin/Heidelberg, Germany, 1995; pp. 86–97. [[CrossRef](#)]
26. Wei, M.; Sun, B.; Wu, W.; Jing, B.B. A multiple objective optimization model for flight arrival and departure scheduling on multiple runways. *Math. Biosci. Eng.* **2020**, *17*, 5545–5560. [[CrossRef](#)] [[PubMed](#)]
27. Hu, X.B.; Di Paolo, E. Binary-representation-based genetic algorithm for flight arrival sequencing and scheduling. *IEEE Trans. Intell. Transp. Syst.* **2008**, *9*, 301–310. [[CrossRef](#)]
28. Androutsopoulos, K.N.; Manousakis, E.G.; Madas, M.A. Modeling and solving a bi-objective airport slot scheduling problem. *Eur. J. Oper. Res.* **2020**, *284*, 135–151. [[CrossRef](#)]
29. Cai, K.; Li, Z.; Guo, T.; Du, W. Multiairport departure scheduling via multiagent reinforcement learning. *IEEE Intell. Transp. Syst. Mag.* **2023**, *16*, 102–116. [[CrossRef](#)]
30. Sun, Y.; Wang, S.; Feng, S.; Ding, S.; Pang, C.; Shang, J.; Liu, J.; Chen, X.; Zhao, Y.; Lu, Y.; et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv* **2021**, arXiv:2107.02137. [[CrossRef](#)]
31. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. LLaMa 2: Open foundation and fine-tuned chat models. *arXiv* **2023**, arXiv:2307.09288. [[CrossRef](#)]
32. Mao, J.; Qian, Y.; Ye, J.; Zhao, H.; Wang, Y. Gpt-driver: Learning to drive with gpt. *arXiv* **2023**, arXiv:2310.01415. [[CrossRef](#)]
33. Thirunavukarasu, A.J.; Ting, D.S.J.; Elangovan, K.; Gutierrez, L.; Tan, T.F.; Ting, D.S.W. Large language models in medicine. *Nat. Med.* **2023**, *29*, 1930–1940. [[CrossRef](#)] [[PubMed](#)]
34. Zhang, Q.; Mott, J.H. An Exploratory Assessment of LLM’s Potential Toward Flight Trajectory Reconstruction Analysis. *arXiv* **2024**, arXiv:2401.06204. [[CrossRef](#)]
35. Abdulhak, S.; Hubbard, W.; Gopalakrishnan, K.; Li, M.Z. CHATATC: Large Language Model-Driven Conversational Agents for Supporting Strategic Air Traffic Flow Management. *arXiv* **2024**, arXiv:2402.14850. [[CrossRef](#)]
36. Thatcher, J.; Amankhan, A.; Eldred, M.; Suboyin, A.; Sonne-Schmidt, C.; Rehman, A. Clear Skies Ahead: Optimizing Operations Through Large Language Models and AI to Reduce Emissions and Costs for a Regional NOC. In Proceedings of the International Petroleum Technology Conference, Dhahran, Saudi Arabia, 12 February 2024. [[CrossRef](#)]

37. Wang, L.; Chou, J.; Tien, A.; Zhou, X.; Baumgartner, D. AviationGPT: A large language model for the aviation domain. In Proceedings of the Aiaa Aviation Forum and Ascend 2024, Las Vegas, NV, USA, 29 July–2 August 2024; p. 4250.
38. Khan, W.A.; Chung, S.-H.; Eltoukhy, A.E.E.; Khurshid, F. A novel parallel series data-driven model for IATA-coded flight delays prediction and features analysis. *J. Air Transp. Manag.* **2024**, *114*, 102488. [[CrossRef](#)]
39. Wen, X.; Chung, S.-H.; Ma, H.-L.; Khan, W.A. Airline crew scheduling with sustainability enhancement by data analytics under circular economy. *Ann. Oper. Res.* **2023**, 1–27 [[CrossRef](#)]
40. Khan, W.A.; Chung, S.-H.; Ma, H.-L.; Liu, S.Q.; Chan, C.Y. A novel self-organizing constructive neural network for estimating aircraft trip fuel consumption. *Transp. Res. Part E Logist. Transp. Rev.* **2019**, *132*, 72–96. [[CrossRef](#)]
41. Qin, Y.; Ma, H.-M.; Chan, F.T.S.; Khan, W.A. A scenario-based stochastic programming approach for aircraft expendable and rotatable spare parts planning in MRO provider. *Ind. Manag. Data Syst.* **2020**, *120*, 1635–1657. [[CrossRef](#)]

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.