aerospace

MDPI

Article

Adaptive Dynamic Programming with Reinforcement Learning on Optimization of Flight Departure Scheduling

Hong Liu [1,2,*], Song Li [2], Fang Sun [3], Wei Fan [4], Wai-Hung Ip [5] and Kai-Leung Yung [5]

[1] Department of Air Traffic Management, Civil Aviation University of China, Tianjin 300300, China
[2] State Key Laboratory of Air Traffic Management System, Nanjing 210014, China; lisong12@cetc.com.cn
[3] Department of Science, Civil Aviation University of China, Tianjin 300300, China; fsun@cauc.edu.cn
[4] Department of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China; wfan@cauc.edu.cn
[5] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China; wh.ip@polyu.edu.hk (W.-H.I.); kl.yung@polyu.edu.hk (K.-L.Y.)
* Correspondence: h_liu@cauc.edu.cn

Abstract: The intricacies of air traffic departure scheduling, especially when numerous flights are delayed, frequently impede the implementation of automated decision-making for scheduling. To surmount this obstacle, a mathematical model is proposed, and a dynamic simulation framework is designed to tackle the scheduling dilemma. An optimization control strategy is based on adaptive dynamic programming (ADP), focusing on minimizing the cumulative delay time for a cohort of delayed aircraft amidst congestion. This technique harnesses an approximation of the dynamic programming value function, augmented by reinforcement learning to enhance the approximation and alleviate the computational complexity as the number of flights increases. Comparative analyses with alternative approaches, including the branch and bound algorithm for static conditions and the first-come, first-served (FCFS) algorithm for routine scenarios, are conducted. Moreover, perturbation simulations of ADP parameters validate the method's robustness and efficacy. ADP, when integrated with reinforcement learning, demonstrates time efficiency and reliability, positioning it as a viable solution for decision-making in departure management systems.

Keywords: adaptive dynamic programming; departure scheduling; constrained integer optimization; system robustness

Citation: Liu, H.; Li, S.; Sun, F.; Fan, W.; Ip, W.-H.; Yung, K.-L. Adaptive Dynamic Programming with Reinforcement Learning on Optimization of Flight Departure Scheduling. *Aerospace* **2024**, *11*, 754. https://doi.org/10.3390/aerospace11090754

Academic Editor: Sebastian Schier-Morgenthal

Received: 1 July 2024
Revised: 31 August 2024
Accepted: 5 September 2024
Published: 13 September 2024

1. Introduction

Air traffic management, particularly the scheduling of flight departures, is a complex and multifaceted problem that has significant implications for both airline operations and passenger satisfaction. This complexity becomes even more pronounced during periods of extensive flight delays, which can cascade across the entire network, exacerbating congestion and inefficiencies. In response to the challenges posed by air traffic congestion, significant strides have been taken to mitigate the effects of flight delays. Nevertheless, the unpredictability introduced by various elements—such as airport ground support operations, airline policies, crew dynamics, and passenger behavior—continues to disrupt the predictability of flight schedules and the consistency of control decisions. Streamlining the arrival and departure sequencing process is a proven strategy to alleviate the strain caused by congestion and enhance overall operational efficiency.

Arrival scheduling has seen extensive research, with Dear [1] pioneering position shift constraints, Psaraftis [2] and Bianco [3] formulating it as a cumulative asymmetric travelling salesman problem, and Beasley [4] et al. proposing mixed integer programming solutions. Compared with arrival scheduling, departure scheduling allows airlines and air traffic controllers to manage flights from the outset, which is crucial for preventing delays. By optimizing the departure sequence, potential delays can be addressed before they

propagate through the system, reducing the likelihood of cascading effects that typically occur with arrival delays. Departure scheduling refers to the challenge of determining the optimal sequence and time slots of departing aircraft from an airport to minimize delays and maximize efficiency. It involves coordinating the release of multiple aircraft while considering various constraints, such as air traffic control regulations, required time separation between consecutive departures, and potential disruptions like weather conditions or airspace congestion. The primary objective is to ensure a smooth and efficient flow of air traffic while maintaining safety and adhering to operational restrictions.

Efficient departure scheduling can lead to significant cost savings in terms of fuel, crew time, and operational efficiency. Bolender [5] first proposed that the core of flight departure management consists of the scheduling of aircraft for departure and merging departure flights onto their field routes. Atkin et al. [6] investigated the problem of runway scheduling at London Heathrow Airport, aiming to minimize the total departure time for a set of aircraft. Their study was based on both the greedy algorithm and the genetic algorithm. Avella [7] et al. set up an integer programming model to consider the scheduling problem under the interference of arrival flights. Ma et al. [8,9] proposed a mixed integer programming model for departure runway scheduling and solved the model by using the simulated annealing algorithm. Bikir et al. [10] proposed a genetic algorithm-based method to help air traffic controllers organize the departure sequence according to the standard instrument departure (SID) configuration. The departure scheduling problem is highly complex, as it involves numerous constraints, variables, and state spaces, making it computationally intensive and challenging to solve, often classified as NP-hard.

ADP has emerged as a powerful technique that transcends the limitations of traditional dynamic programming while retaining its core principles of dynamic control, as highlighted by Wang et al. [11]. This method employs a function approximation framework to estimate the performance index function, thereby enabling the derivation of an approximate optimal control policy in line with the principle of optimality. As an intelligent control strategy endowed with learning and optimization capabilities, ADP has garnered significant attention in the realms of control theory and computational intelligence. Its potential for addressing optimal control challenges is vast, encompassing applications in diverse areas such as motion control, optimization of operating room and nursing unit scheduling, storage assignment problems, microservice composition optimization, solar energy forecasting, and satellite range scheduling. Pioneering works in this field include contributions by Fung et al. [12], Chiang et al. [13], Tu et al. [14], Xhafa et al. [15], Gao et al. [16] and Chang et al. [17].

Moreover, the integration of ADP with model predictive control (MPC) has been explored by Greatwood [18] and Zhang [19], enhancing the predictive capabilities of control systems, particularly for data-driven applications. This synergy between ADP and MPC offers a robust approach to control, optimizing system performance while adapting to the dynamic and often unpredictable nature of real-world environments.

The applicability of ADP extends to traffic flow control, where it has been increasingly utilized to manage and optimize road traffic conditions. Cai et al. [20] introduced an adaptive traffic signal controller designed for real-time operations, leveraging ADP to significantly reduce computational demands and alleviate traffic congestion. Yin et al. [21] proposed a vehicle-following model within a distributed traffic network, demonstrating that ADP outperforms other control methods across various performance metrics.

While previous studies have primarily focused on road transportation systems, this paper explores air traffic, specifically addressing the departure scheduling problem through the application of ADP. To our knowledge, there have been few attempts to address the scheduling of departing flights using ADP. This paper makes a significant contribution by introducing a dynamic programming model with reinforcement learning tailored to the departure scheduling problem, followed by the design and implementation of an ADP-based solution. The proposed approach stands apart from existing models by incorporating the impact of random disturbances in flight release, such as those caused by weather

conditions and air traffic control constraints. Actual release times for flights are determined by optimizing flight scheduling while accounting for these random delays. The goal of the dynamic programming model is to minimize the total delay time for a group of aircraft held due to congestion, while also considering the necessary time separation between consecutive aircraft. This paper demonstrates that an adaptive departure scheduling controller, leveraging both ADP and reinforcement learning, can significantly reduce flight delays while maintaining high computational efficiency.

This paper is structured as follows. Section 2 presents the model and its mathematical formulation for departure scheduling. As the state space expands, the computational demand of dynamic programming escalates exponentially. Section 3 outlines the basics of the ADP approach, highlighting the flexibility of its open framework for incorporating diverse learning methods. Temporal-difference learning, discussed in detail, is utilized in the model. Section 4 details the numerical experiments and results. The paper concludes in Section 5 with insights on potential future advancements.

2. Mathematical Model for Departure Scheduling

In this section, the fundamentals of the dynamic programming model are presented. Initially, the basics of departure scheduling are discussed in Section 2.1. Subsequently, a dynamic programming model for determining the departure sequence is introduced in Section 2.2. The approach is designed to systematically address the complexities inherent in departure scheduling.

2.1. Fundamentals of Departure Scheduling

Under typical conditions, flights are dispatched according to their schedules. Flight crews request departure clearance based on the anticipated departure time, and air traffic controllers issue clearances adhering to the first-come, first-served principle. However, delays are often necessitated by severe weather or air traffic flow management, leading to adjustments and reordering of departure times. After being reordered by a departure optimizer, the flight follows the departure procedures, including pushback and startup, taxiing, queuing near the runway, lining up, and taking off. During this process, various control strategies, such as sequencing, holding, and conflict detection and resolution, are applied to ensure a safe, orderly, and efficient traffic flow. The departure optimizer is typically influenced by factors such as objectives, constraints, random factors like weather-related uncertainties, flow control, operational restrictions, and disturbances caused by control strategies applied to terminal traffic. In this paper, the departure optimizer is designed using ADP combined with reinforcement learning techniques. Figure 1 illustrates this process, with the departure optimizer highlighted in the blue box. This paper aims to establish an optimal departure sequence with the goal of minimizing total delays. For departure scheduling, constraints inherent to the terminal area system must be adhered to.

(1) Minimum Takeoff Separation Constraints (MTSC). In the interest of departure safety, a critical factor to consider is the minimum takeoff separation between consecutive aircraft originating from the same airport. This measure is crucial for avoiding the hazards of wake vortex, which are the turbulent air currents generated by an aircraft's wings as it ascends.

The separation criteria are generally based on the aircraft's total takeoff weight, providing a straightforward yet effective method for determining the necessary distance between consecutive takeoffs. Heavier aircraft, which produce more potent wake vortex, require a greater separation distance compared to lighter aircraft, which necessitate a smaller separation interval. The International Civil Aviation Organization (ICAO) has set forth recommended minimum separation and the Civil Aviation Authority of China has applied this separation. It is given in terms of distance and was converted into time-based separation using average runway occupancy times, as depicted in Table 1.

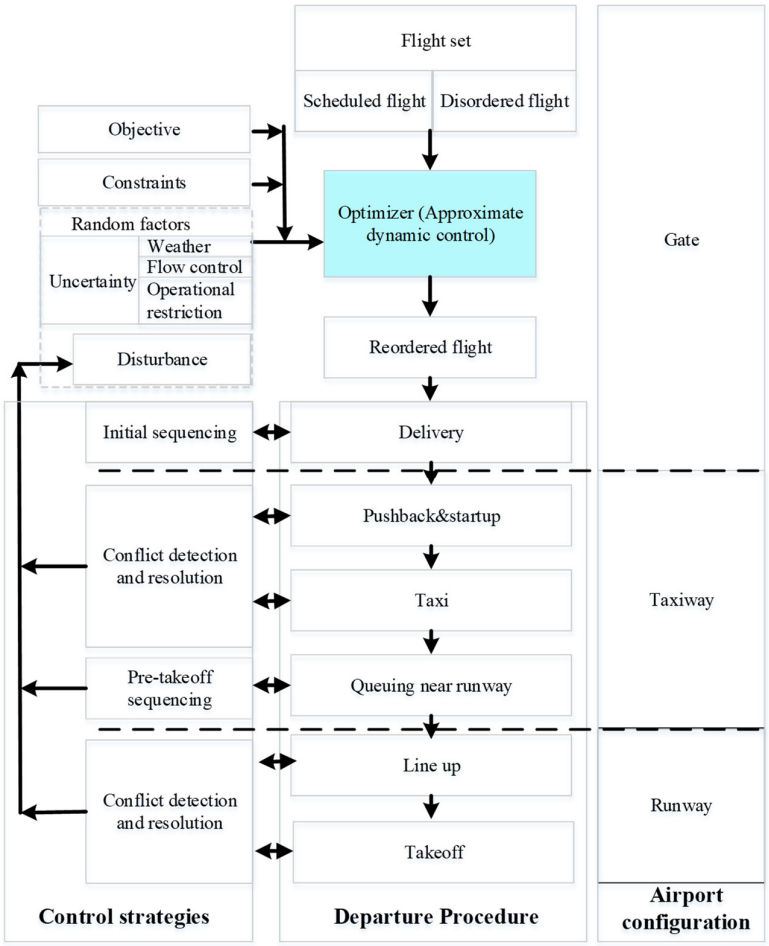

Figure 1. Departure scheduling framework.

Table 1. Average runway occupancy times.

Aircraft Type	Minimum Time (in Seconds)		
	Small (Later)	**Large (Later)**	**Heavy (Later)**
Small (Former)	98	74	74
Large (Former)	138	74	74
Heavy (Former)	167	114	94

(2) Maximum Position Shifting (MPS). The traditional method for sequencing departure flights is the first-come-first-served (FCFS) approach, which prioritizes aircraft based on the order they join the departure queue. While this method is straightforward, it can lead to an exponential increase in the number of possible arrangements as the number of flights (N) grows, with N factorial (N!) permutations available.

To mitigate the heavy workload associated with managing such a vast array of sequencing options, the concept of MPS is employed. MPS is a regulatory parameter that restricts how much an aircraft can deviate from its initial FCFS position, thereby simplifying the sequencing process for air traffic controllers. By defining an MPS, the air traffic controller seeks to strike a balance between minimizing delays and reducing the cognitive load on controllers. This constraint not only enhances operational efficiency but also ensures that the sequencing of departures remains manageable and safe.

2.2. Dynamic Programming

In this subsection, we propose a dynamic model to determine departure sequence, complying with separation rules. The dynamic programming model for departure is based on Equations (1)–(5) with the objective of minimizing the delay measured in time.

Let $FL = (1, 2, \ldots n)^T$ be the column vector of departure flight indices, where N is the total number of flights under consideration. The process of solving departure sequencing is properly divided into N interrelated stages. For $i \in FL$, denote $\Delta t^d(i)$, the time span required for flight i departing from the airport, obtained by calculating the difference between the time obtaining air traffic clearance to the time vacating the runway. Let $\Delta t = (\Delta t(1), \Delta t(2), \ldots, \Delta t(n))^T$ be the column vector of time, where $\Delta t(i) = \Delta t^d(i) + \theta(i)$ and $\theta(i)$ represent the maximum time-based separation for the safety of the later flight. For example, given the flight i is a small type, then $\theta(i) = 98$ if it is followed by a small-type aircraft, as shown in Table 1.

Denote x as a vector of the system state and $J(x)$ as the true value function associated with state x. Let $g(\cdot)$ be a one-step cost function and α be a discount factor. Generally, a dynamic programming algorithm is associated with the state variable $x \in X$ and the decision variable $u \in U$. Given the initial state x_0 and a sequence of decision u_t, dynamic programming is to solve

$$\min_{u_t \in U} E_{w_t} \left\{ \sum_{t=0}^{T-1} \alpha^t g(x_t, x_{t+1}) \middle| x_0 \right\}. \tag{1}$$

By Bellman's principle, the above equation can be recursively calculated through the following iteration. For $t = N-1, N-2, \ldots, 0$,

$$J^*(x_t) = \min_{u_t \in U} E_{w_t} \left\{ g(x_t, x_{t+1}) + \alpha J^*(x_{t+1} | x_t) \right\}. \tag{2}$$

For our model, step t is a discrete time in which an aircraft is dispatched. w_t expresses the extra delay time due to various random events in step t, including weather conditions, air traffic control, and arrival flight separation.

Denote $T_t = (T_t(1), T_t(2), \ldots, T_t(N))^T$, where $T_t(i)$ is delay of flight $i \in FL$ at step t. For each step t, let s_t be the flight's dispatched status, to be more clear, $s_t = (s_t(1), s_t(2), \ldots, s_t(N))$, where $s_t(i)$ is a binary variable depending on the flight condition, such that for $i \in FL$,

$$s_t(i) = \begin{cases} 1, & \text{if flight } i \text{ waits for being dispatched;} \\ 0, & \text{if flight } i \text{ has been dispatched.} \end{cases}$$

A state x_t can be represented by a combination of delay at T_t and flight status vector s_t. Furthermore, we define $u_t = (u_t(1), u_t(2), \ldots, u_t(N))$ as a decision variable vector, where $u_t(i) = 1$ if the flight i is dispatched at step t and $u_t(l) = 0$ for $l \neq i$ since exactly one flight can be dispatched at one step. In short,

$$u_t(i) = \begin{cases} 1, & \text{if flight } i \text{ is dispatched at step } t; \\ 0, & \text{otherwise.} \end{cases}$$

Under this condition, $s_{t+1}(i) = 0$, and other flight statuses are unchanged. Therefore, the flight status vector can be transferred during increment from t to $t+1$ as

$$s_{t+1} = (s_t + u_t) \bmod 2. \tag{3}$$

The delay time of all the flights that are waiting to be released will increase $\Delta t(i) + w_t$ at step $t+1$ when flight i is dispatched at step t, and the delay time is not increased for the flights having been dispatched. So, the transition of delay time state equation is given as

$$T_{t+1} = T_t + \left({u_t}^T \Delta t + w_t \right) \mathrm{s}_{t+1}. \tag{4}$$

The one-step cost function g is defined on the objective. Let $c = (c(1), c(2), \ldots, c(N))^T$ be the unit delay cost of the flight. Then g can be defined as follows when the objective is to minimize the delay cost:

$$g(x_t, x_{t+1}) = \left({u_t}^T \Delta t + w_t\right) \mathrm{C}_{t+1}^T I, \tag{5}$$

where C_{t+1} is an N dimensional column vector, $C_{t+1}(i) = s_{t+1}(i)c(i)$, for $i \in FL$, I is an N dimensional column vector, and $I(i) = 1$, for $i \in FL$. Equation (5) describes the total delay time of the flight waiting for releasing from x_t to x_{t+1}.

3. Adaptive Dynamic Programming

ADP is an approximate technique in dynamic programming. It employs function approximation methods, like neural networks or fuzzy controllers, to closely approximate the cost function, aiming to optimize the entire system's control. Werbos et al. [10] have shown that the adaptive method can theoretically approximate solutions to any control or planning problem framed as an optimization issue. In this paper, ADP is applied to approximate the true function for departure scheduling, enhancing decision-making in this domain.

3.1. Fundamentals of Adaptive Dynamic Programming

The fundamentals of ADP can be sketched as follows: by using an approximate value function, the dynamic programming problem is solved. The value iteration takes a process stepping forward into time. In each iteration step, the approximation to the true value function is updated on each observation of state transition with the aid of reinforcement learning. The approximate function and the reinforcement learning are two key essentials.

To form a mathematical formulation for the ADP approach, we explore a continuous approximation function

$$\tilde{J}^*(\cdot, r) : X \times R^k \to R. \tag{6}$$

Equation (6) approximate the true value function $J(\cdot) : X \to R$, where r is the parameter vector. At each step t, compared to Equation (2), the following equation is calculated for $t = 0, 1, \ldots, N-1$.

$$\tilde{J}^*(x_t) = \min_{u_t \in U} E_{w_t}\left\{ g(x_t, x_{t+1}) + \alpha \tilde{J}^*(x_{t+1}, r_t) \right\}, \tag{7}$$

and the control strategy is obtained by

$$u^* = \arg\min_{u_t \in U} E_{w_t}\left\{ g(x_t, x_{t+1}) + \alpha \tilde{J}^*(x_{t+1}, r_t) \right\}. \tag{8}$$

In the optimization value iteration, the optimal value function and the optimal control strategy are approximated, respectively, through (7) and (8) based on the actual flight dispatch status. Equation (7) is calculated by visiting the actual state x_t, so the computation burden is strongly reduced.

In terms of the approximate function, there are lots of techniques available, such as aggregation, regression, and most frequently, neural networks. For simplicity, Cai's [20] model of adaptive traffic signal control is referred. Dynamic programming is approximated by a continuous linear approximation function as follows,

$$\tilde{J}(x, r) = \sum_{k=1}^{N} r^T(k)\phi_k(x), \tag{9}$$

where $\phi_k(x)$ is a feature-extraction function, which can be expressed as follows,

$$\phi_k(i) = \begin{cases} \binom{D_k(i)}{0}, \ if \ s(i) = 1; \\ \binom{0}{D_k(i)}, if \ s(i) = 0, \end{cases}$$

where $D_k(i)$ is the delay time accumulated till step t for aircraft i. The parameter r is a two-dimensional vector, which is expressed as

$$r(i) = \begin{pmatrix} r^+(i) \\ r^-(i) \end{pmatrix}.$$

In the next subsection, the reinforcement learning algorithm is chosen for supervising the update of the parameter r.

3.2. Reinforcement Learning

In the standard framework for reinforcement learning, a learning agent automatically determines the ideal behavior after observing the states of its environment. Simple reward feedback is required for the agent to learn its behavior, and this is known as the reinforcement signal. The agent must learn to choose actions to maximize a long-term sum or average of the future payoffs it will receive.

In this paper, reinforcement learning techniques exploit an ADP model to update the functional parameters of the approximation function shown in (9) upon each observation of a state transition. The objective is to improve approximations as more state transitions are observed. Many different algorithms address this issue, such as the Monte Carlo algorithm, temporal-difference learning, and so on. In this context, the approach of Cai et al. [20] is followed, applying the temporal-difference learning algorithm originally proposed by Sutton et al. [22].

For the problem of improving approximations in (7), the parameters are updated according to the formula

$$r_{t+1} = r_t + \eta_t d_t \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \phi_k(x), \tag{10}$$

where d_t is the temporal difference defined as

$$d_t = g(x_t, x_{t+1}) + \alpha \widetilde{J}(x_{t+1}, r_t) - \widetilde{J}(x_t, r_t),$$

η_t is a sequence of scalar step sizes satisfying the following terms for convergence

$$\sum_{t=0}^{\infty} \eta_t = \infty, \tag{11}$$

and

$$\sum_{t=0}^{\infty} \eta_t^2 = \infty. \tag{12}$$

Note that the scalar sequence could still be properly selected even though (11) and (12) are not satisfied if the iteration terminates in finite steps. Parameter λ takes value in interval [0, 1], which is known as the trace eligibility factor.

3.3. Flight Scheduling Algorithm

The flight scheduling algorithm procedure based on ADP is presented in the following.

Step 1: Initialization: set $t = 0$, input the initial state of the flights x_0, T_0, s_0,r_0, the initial value function $\widetilde{J}(x_0)$ and learning rate η_0. It is clear that $s_0 = (1, 1, \ldots, 1)^T$ and T_0 is the initial delay time of the flights.

Step 2: Solve (8) and (7) under the constraint of MPS to obtain u_t^* and $\widetilde{J}_t^*(x_t)$.

Step 3: Reception of random information: obtain the new extra delay time w_t;

Step 4: Implement the optimal decision and transfer the system state through (3) and (4);

Step 5: Update the approximate value function using (10);

Step 6: Go back to step 2 for $t < N$, and stop otherwise.

4. Simulation Framework and Results

The simulation is performed on a PC configured with Intel(R) Core (TM) i7-4790 CPU 3.60 GHz. The test environment is built on MATLAB R2016a, and the MATLAB built-in function 'rand' is called to initialize airplane types, while the delay time is subjected to the power law distribution, as shown in Figure 2. In Figure 3, the horizontal axis represents delay time, while the vertical axis shows the ratio of delayed departure flights to the total number of flights. From Figures 2 and 3, the delay time of most flights is less than 30 min.

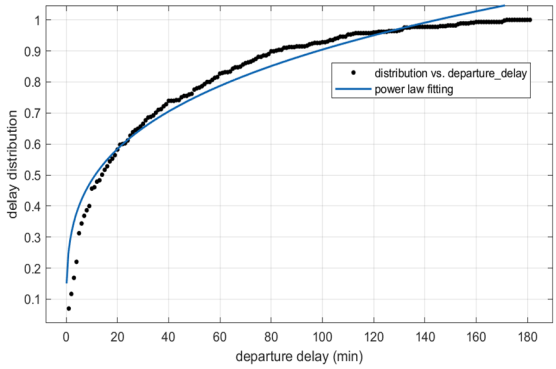

Figure 2. The power law distribution of delay time.

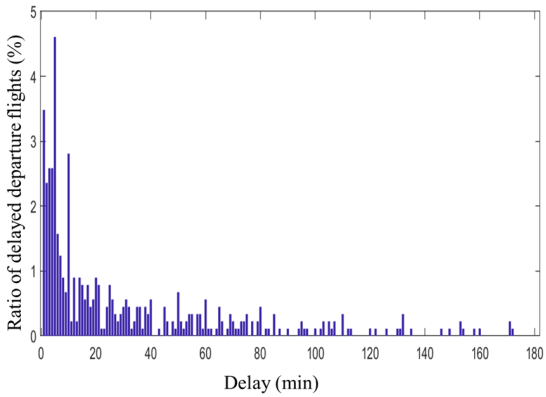

Figure 3. The ratio of delayed departure flights.

In order to study the performance of the algorithm, two scenarios are designed. Both scenarios consider the initial delayed time and aircraft type as input. In the first scenario, time complexity and optimization performance of the ADP algorithm is studied, while the later scenario mainly focuses on the perturbation of the ADP algorithm relative to its initial parameters.

4.1. Performance Simulation Scenario

To analyze the time performance of the ADP algorithm, the number of flights is varied from 10 to 1000, increasing in increments of 10. In order to study the optimization performance of the ADP algorithm, the branch and bound (BNB) algorithm is modeled for comparison. However, the comparison is performed under a static scenario without

considering stochastic factors. Furthermore, the exponential time complexity of BNB makes the running time of BNB unacceptable when the number of flights increases greater than 10.

BNB is based on the integer nonlinear programming model, stated as follows. The objective function is to minimize the total delay time of all flights

$$min \sum_{i=0}^{N} \sum_{j=0}^{N} \left(a_{i,j} + p_{i,j}\right) \sum_{k=1}^{N-1} x_{i,k} x_{j,k+1}, \tag{13}$$

where N is the total number of delayed flights, $a_{i,j}$ is the time separation consumed if the jth flight is delivered behind the ith flight, and $p_{i,j}$ is its separation minimum, binary decision variable $x_{i,k} = 1$ if the ith flight is delivered at the kth time and $x_{i,k} = 0$ otherwise.

The constraints include

$$\sum_{i=0}^{N} x_{i,j} = 1, \text{ for all } 0 \leq j \leq N, \tag{14}$$

$$\sum_{j=0}^{N} x_{i,j} = 1, \text{ for all } 0 \leq i \leq N, \tag{15}$$

$$\left|x_{i,j} - x_{i,j}^{0}\right| \leq \zeta, \text{ for } all\ 0 \leq i, j \leq N \tag{16}$$

$$x_{i,j} \in \{0,1\}, 0 \leq i, j \leq N, \tag{17}$$

where the constraint (14) implies that at each time, exactly one flight can be put into the delivery sequence; the second constraint (15) represents that each flight can be delivered only once; constraint (16) is the MPS for the given initial delivery sequence $x_{i,j}^{0}$ and the maximum position shifting ζ. Equation (17) shows the binary decision variable $x_{i,j}$.

4.2. Dynamic Process and Perturbation Simulation Scenario

Figure 4 illustrates the dynamic process at step t for making the decision regarding u_t and calculating the total delay time T_t by accounting for the random delay time w_t. It is clear that s_t represents the departing status at step t and d_t denotes the temporal difference at the end of step t. The decision variable u_t is made at the end of step t to decide whether the control state will change. The controller state s_{t+1} is then determined by s_t and the decision variable u_t.

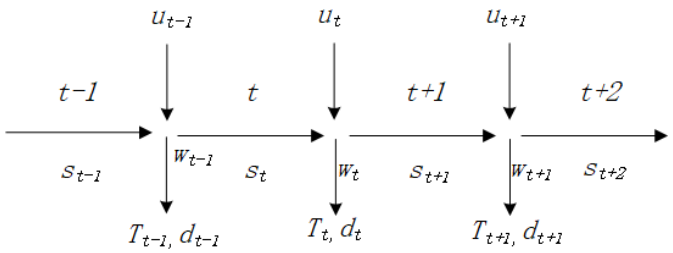

Figure 4. The process of the state transition.

The perturbation simulation scenario is designed to test the sensibility of the trace eligibility factor λ, the scalar sequence η_t, and the discount factor α. Without loss of generality, the number of flights is set to be 200. To ensure general applicability, the number of flights is set to 200. Initial delay times are randomly generated, ranging from 30 to 60 min, following a pseudo-uniform distribution.

4.3. Simulation Results and Analysis

The airport configuration consists of three parts: apron, taxiway, and departure runway. Four vertical taxiways connect the apron with one parallel taxiway. This configuration is

a classical design for most airports in China. Only one departure runway is selected, as shown in Figure 5.

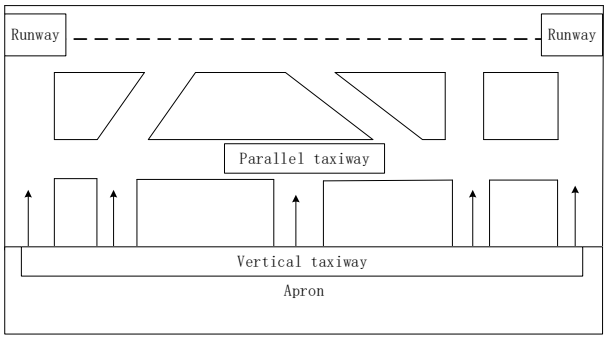

Figure 5. Simulation of airport configuration.

In this subsection, the performance of ADP is analyzed. Firstly, a comparison of ADP with BNB is illustrated. Secondly, the optimization results of ADP are compared with the optimization results of the FCFS order. At last, the sensitivities of ADP to the trace eligibility factor λ, discount factor α, and scalar sequence η are given, respectively.

In order to test the time performance of ADP, thousands of test samples are generated based on the flight number. The aircraft types of the sample data obey uniform distribution. Figure 6 illustrates the time performance of ADP and BNB. In this case, the flight number N increases from 4 to 8, as shown in the horizontal coordination. For each flight number N, the average run time of the test samples is collected and given in the vertical coordination.

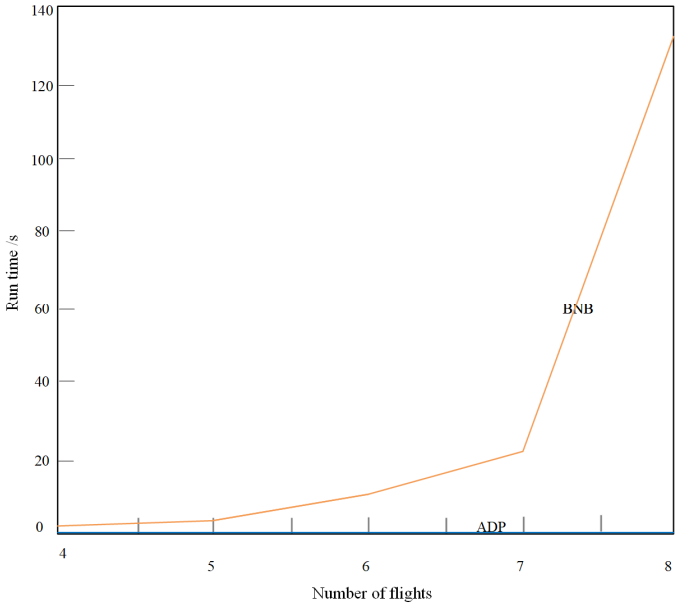

Figure 6. Run time comparison between BNB and ADP in small scale.

It is easily observed that the running time of BNB assumes the geometric series growth while the ADP is linear growth relative to the flight number. The BNB algorithm takes several hours for solving each test sample, as the flight number N equals 9. The running time is too long to be accepted for application of the BNB algorithm in solving the test sample if the flight number N is greater than 10.

Figure 7 shows the run time of ADP compared to FCFS, as the flight number N increases from 50 to 200. Both the run time of ADP and the run time of FCFS increase linearly relative to the flight number N, whose gradient coefficients are approximate to

0.0032 and 0.00087, respectively. It could be observed that both the maximum run time of the ADP and the maximum run time of the FCFS are less than 1 s.

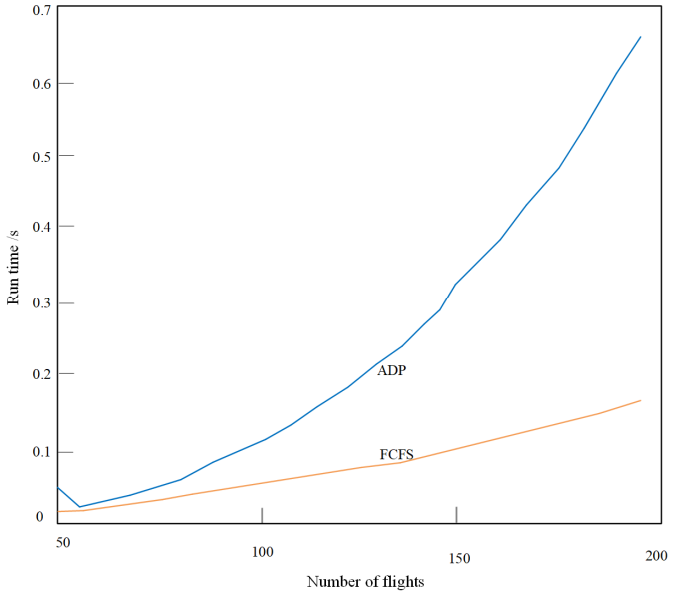

Figure 7. Run time comparison between ADP and FCFS in large scale.

The optimization performance of ADP is tested under two scenarios. One scenario, namely small-scale scenario, is based on sample data with the flight number $N = 4, 5, 6, 7$. The other scenario, namely large-scale scenario, is based on sample data with the flight number n increasing from 50 to 200. The small-scale scenario test result is shown in Figure 5. In this scenario, the influence of different aircraft types to the total delay time is considered in Figure 8; the x-axis is marked with symbol nX, where n is the flight number, and the suffix X represents the ratio of the aircraft type. The suffix X takes two values E and S, where E represents the ratio of heavy aircraft type to large aircraft type and to small aircraft type, which equals 1:1:1, while the suffix S represents the ratio of heavy aircraft type to large aircraft type and to small aircraft type, which equals 1:2:1. The vertical axis marked by the y-axis represents the total delay time. The total delay time under ADP improves more than 20% compared to the total delay time under FCFS, while the difference between the total delay time under ADP and the optimum value of the total delay time under BNB is kept within 10%.

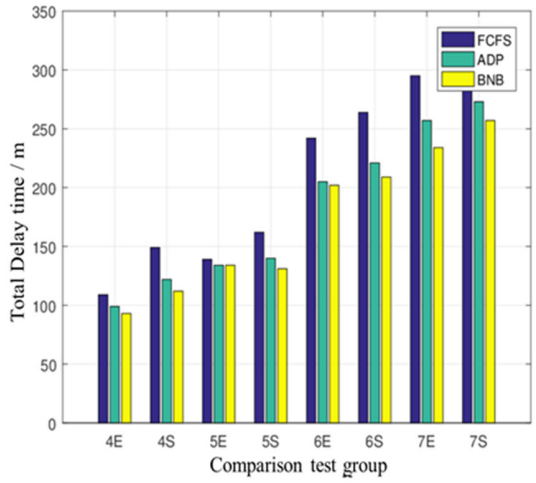

Figure 8. Optimization comparison of ADP, BNB and FCFS in small scale.

The large-scale scenario test result is shown in Figure 9. It could be observed that the objective value of total delay increased as the number of flights increases for both ADP and FCFS. The maximum improvement for the objective value of the ADP compared to that of the FCFS is nearly 2000 min, and the average relative improvement ratio of ADP compared to the FCFS exceeds 34%.

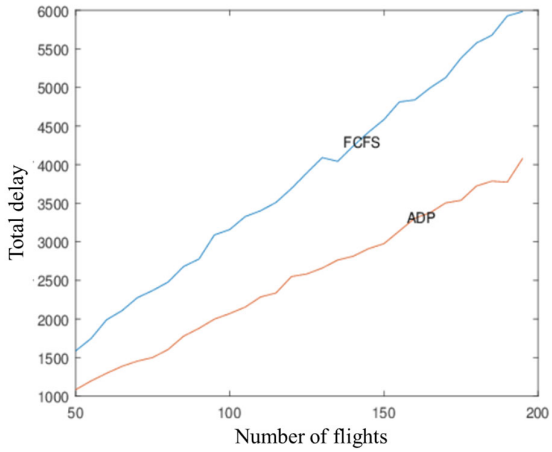

Figure 9. Optimization comparison of ADP and FCFS.

In the final part, a perturbation analysis is conducted. First, the scalar sequence η is fixed, and the trend of the optimization value is observed relative to changes in the trace eligibility factor λ and discount factor α.

The results are shown in Figure 10. It can be seen that the optimization value remains stable, with significant changes in λ and α resulting in only minor variations in the optimization value. The maximum delay occurs when λ is in the range of [0.22, 0.24], and the minimum delay occurs when λ is in the range of [0.19, 0.21] and α is in the range of [0.25, 0.26].

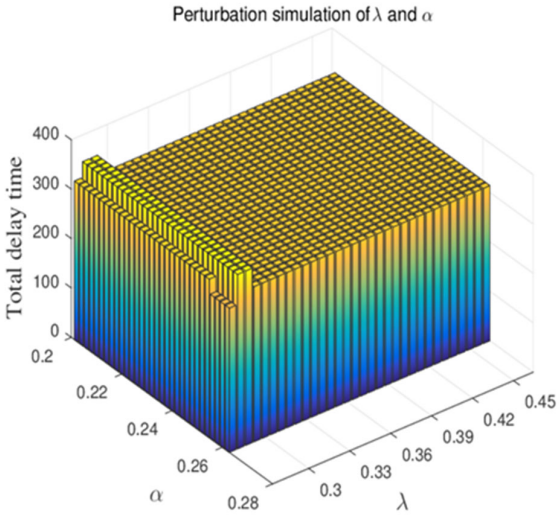

Figure 10. Perturbation analysis of parameter α and λ.

Next, the scalar sequence (η) is varied to test the ADP's sensitivity in this scenario. η is selected to decrease inversely in Equations (18) and (19), following an inverse proportionality relationship and decreases exponentially in Equations (20) and (21).

$$\eta_{t_1} = {t_1}^{-1} \tag{18}$$

$$\eta_{t_2} = 10^{-3} {t_2}^{-1} \tag{19}$$

$$\eta_{t_3} = e^{-t_3} \tag{20}$$

$$\eta_{t_4} = 10^3 * e^{-t_4} \tag{21}$$

The experimental results are illustrated in Figure 11. The influence of scalar sequence is greater on the solver of ADP than the trace eligibility factor and the discount factor. The volatility of the optimization value under the sequence η_{t_3} is the greatest, and the volatility of the optimization value under the sequence η_{t_1} is the smallest.

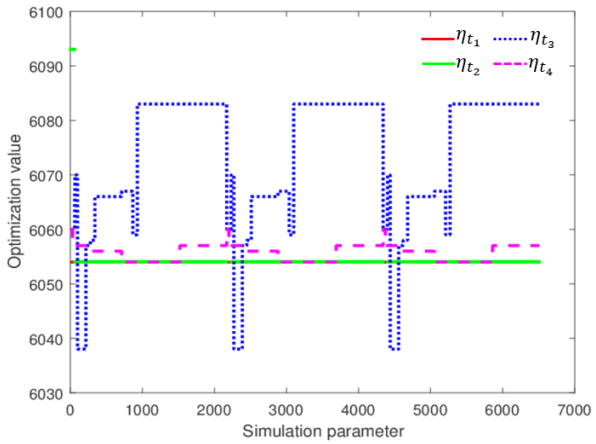

Figure 11. Perturbation analysis of parameter η.

5. Conclusions

In conclusion, a novel deep learning framework, enhanced with reinforcement learning, has been effectively employed for planning and controlling flight departure sequencing. Compared to traditional methods like BNB and FCFS, ADP shows superior performance in runtime efficiency. The solution gap of total delay times is kept within 10% by ADP compared to BNB. While BNB becomes impractical due to its exponential time complexity as the number of flights increases, ADP maintains a linear growth in runtime, making it a viable solution for large airports. Additionally, ADP consistently outperforms FCFS, achieving substantial reductions in total delay times.

Perturbation analysis reveals that the scalar sequence factor has a more significant impact on optimization outcomes compared to both the trace eligibility factor and the discount factor. Exponential changes in the scalar sequence factor can lead to substantial fluctuations in the optimization value, potentially guiding the ADP toward global optimization. Conversely, geometric variations in this factor increase the risk of the ADP system becoming trapped in local optima.

Based on these findings, the generalized ADP with reinforcement learning offers a promising approach for designing algorithms that achieve time efficiency without compromising solution optimality.

In future research, the departure/arrival mixed dynamic test environment incorporating fairness considering airlines and passengers into the system model shall be considered.

Author Contributions: Conceptualization, H.L. and F.S.; methodology, F.S.; software, H.L.; validation, H.L. and F.S.; formal analysis, H.L.; investigation, S.L.; resources, H.L.; data curation, H.L.; writing, H.L.; visualization, H.L.; supervision, W.F.; project administration, W.-H.I. and K.-L.Y.; funding acquisition, H.L., W.-H.I. and K.-L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (52272356); Key Laboratory of Air Traffic Management System and Technology (SKLATM202001); Civil Aviation Safety Capability Building Project (ASSA2023/29); Project of Hong Kong Polytechnic University

(WZOW); The research is also in part supported by the Research Centre of Deep Space Explorations (RCDSE), The Hong Kong Polytechnic University.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dear, R.G. The Dynamic Scheduling of Aircraft in the Near Terminal Area. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1978. [CrossRef]
2. Psaraftis, H.N. A Dynamic Programming Approach to the Aircraft Sequencing Problem. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1978.
3. Bianco, L.; Ricciardelli, S.; Rinaldi, G.; Sassano, A. Scheduling tasks with sequence-dependent processing times. *Nav. Res. Logist.* **1988**, *35*, 177–184. [CrossRef]
4. Beasley, J.E.; Krishnamoorthy, M.; Sharaiha, Y.M.; Abramson, D. Scheduling aircraft landings—The static case. *Transp. Sci.* **2000**, *34*, 180–197. [CrossRef]
5. Bolender, M.A. Scheduling and Control Strategies for the Departure Problem in Air Traffic Control. Ph.D. Thesis, University of Cincinnati, Cincinnati, OH, USA, 2000.
6. Atkin, J.A.D.; Burke, E.K.; Greenwood, J.S.; Reeson, D. Hybrid metaheuristics to aid runway scheduling at London Heathrow airport. *Transp. Sci.* **2007**, *41*, 90–106. [CrossRef]
7. Avella, P.; Boccia, M.; Mannino, C.; Vasilyev, I. Time-indexed formulations for the runway scheduling problem. *Transp. Sci.* **2017**, *51*, 1196–1209. [CrossRef]
8. Ma, J.; Sbihi, M.; Delahaye, D. Optimization of departure runway scheduling incorporating arrival crossing. *Int. Trans. Oper. Res.* **2019**, *28*, 615–637. [CrossRef]
9. Ma, J.; Delahaye, D.; Liang, M. Arrival and departure sequencing, considering runway assignment preferences and crossings. *Aerospace* **2024**, *11*, 604. [CrossRef]
10. Bikir, A.; Idrissi, O.; Mansouri, K.; Qbadou, M. An optimized air traffic departure sequence according to the standard instrument departures. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 1364–1371. [CrossRef]
11. Wang, F.-Y.; Zhang, H.; Liu, D. Adaptive dynamic programming: An introduction. *IEEE Comput. Intell. Mag.* **2009**, *4*, 39–47. [CrossRef]
12. Fung, V.W.; Yung, K.C. An intelligent approach for improving printed circuit board assembly process performance in smart manufacturing. *Int. J. Eng. Bus. Manag.* **2020**, *12*, 184797902094618. [CrossRef]
13. Chiang, A.J.; Jeang, A.; Chiang, P.S.; Chung, C.-P. Multi-objective optimization for simultaneous operating room and nursing unit scheduling. *Int. J. Eng. Bus. Manag.* **2020**, *11*, 184797901989102. [CrossRef]
14. Tu, M.; Yang, M.-F.; Kao, S.-L.; Lin, F.-C.; Wu, M.-H.; Lin, C.-K. Using a heuristic multi-objective genetic algorithm to solve the storage assignment problem for CPS-based pick-and-pass system. *Enterp. Inf. Syst.* **2020**, *15*, 1238–1259. [CrossRef]
15. Xhafa, F.; Ip, A.W. Optimization problems and resolution methods in satellite scheduling and space-craft operation: A survey. *Enterp. Inf. Syst.* **2020**, *15*, 1022–1045. [CrossRef]
16. Gao, M.; Chen, M.; Liu, A.; Ip, W.H.; Yung, K.L. Optimization of microservice composition based on artificial immune algorithm considering fuzziness and user preference. *Spec. Sect. Data Min. Internet Things* **2020**, *8*, 26385–26404. [CrossRef]
17. Chang, J.-F.; Dong, N.; Ip, W.H.; Yung, K.L. An ensemble learning model based on Bayesian model combination for solar energy Prediction. *J. Renew. Sustain. Energy* **2019**, *11*, 043702. [CrossRef]
18. Greatwood, C.; Richards, A.G. Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control. *Auton. Robots* **2019**, *43*, 1681–1693. [CrossRef]
19. Zhang, X.; Liu, J.; Xu, X.; Yu, S.; Chen, H. Robust learning-based predictive control for discrete-time nonlinear systems with unknown dynamics and state constraints. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 7314–7327. [CrossRef]
20. Cai, C.; Wong, C.K.; Heydecker, B.G. Adaptive traffic signal control using approximate dynamic programming. *Transp. Res. Part C* **2009**, *17*, 456–474. [CrossRef]
21. Yin, B.; Dridi, M.; El Moudni, A. Traffic network micro-simulation model and control algorithm based on approximate dynamic programming. *IET Intell. Transp. Syst.* **2016**, *10*, 186–196. [CrossRef]
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018. [CrossRef]