Li, Haonan; Wu, Xu; Ribeiro, Marta; Santos, Bruno; Zheng, Pan

Article

Deep reinforcement learning approach for real-time airport gate assignment

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Li, Haonan; Wu, Xu; Ribeiro, Marta; Santos, Bruno; Zheng, Pan (2025) : Deep reinforcement learning approach for real-time airport gate assignment, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 14, pp. 1-15, https://doi.org/10.1016/j.orp.2025.100338

Operations Research Perspectives 14 (2025) 100338

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Deep reinforcement learning approach for real-time airport gate assignment

Haonan Li [a],*, Xu Wu [b], Marta Ribeiro [a], Bruno Santos [a], Pan Zheng [c]

[a] Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, Zuid-holland, The Netherlands
[b] Beijing Jiaotong University, Beijing, China
[c] Civil Aviation Management Institute of China, Beijing, China

ARTICLE INFO

Keywords:
Airport gate assignment
Deep reinforcement learning
Asynchronous advantage actor-critic
Real-time optimization

ABSTRACT

Assigning aircraft to gates is one of the most important daily decision problems that airport professionals face. The solution to this problem has raised a significant effort, with many researchers tackling many different variants of this problem. However, most existing studies on gate assignment contain only a static perspective without considering possible future disruptions and uncertainties. We bridge this gap by looking at gate assignments as a dynamic decision-making process. This paper presents the Real-time Gate Assignment Problem Solution (REGAPS) algorithm, an innovative method adept at resolving pre-assignment issues and dynamically optimizing gate assignments in real-time at airports through the integration of Deep Reinforcement Learning (DRL). This work represents the first time that DRL is used with real airport data and a configuration containing a large number of flights and gates. The methodology combines a tailored Markov Decision Process (MDP) formulation with the Asynchronous Advantage Actor–Critic (A3C) architecture. Multiple factors, such as flight schedules, gate availability, and passenger walking time, are considered. An empirical case study demonstrates that the REGAPS outperforms two classic deep Q-learning algorithms and a traditional Genetic Algorithm in terms of reducing passenger walking time and apron gate assignment. Finally, supplementary experiments highlight REGAPS's adaptability under various gate assignment rules for international and domestic flights. The finding demonstrates that not only did REGAPS outperform COVID restrictions, but it can also produce considerable benefits under other policies.

1. Introduction

One of the most significant current discussions in airport operation discipline is gate assignment, the process of allocating flights to airport gates. Based on the statistical data acquired by IATA [1], in 198 of 354 airports surveyed, they lack the capacity to fulfill the existing demand, necessitating a greater degree of flexibility in the assignment of gates. Thus, it is vital to analyze the nature of the gate assignment problem (GAP) and design an efficient and effective approach to ease the stress of airport coordination. Many research works have tackled gate assignment [2]. One key issue with most of the past studies is that they were conducted from a static perspective, assuming a static environment without any unpredictable changes. As a result, no last-minute disturbances are taken into account. However, existing research shows that flight delays negatively affect the performance of static gate assignments [3]. Consequently, these solutions can only be used as strategic planning, from which airport professionals must deviate in real-time to accommodate unplanned changes.

In practice, airport professionals assign gates in two stages: first in pre-assignment and then in real-time. In the pre-assignment phase, gates are assigned in advance based on previous data. As plane travel is so readily disrupted, the pre-assignment plan will almost certainly diverge from the final plan. Operators often handle disturbances manually, depending on their experience. Some researchers observed the deviation and attempted to solve it [4–6]. These attempts to mitigate the impact of disruption by strengthening the assignment plan's robustness or researching solutions to the reassignment challenge. However, current systems are incapable of rapidly adjusting the assignment plan in response to changes in the flight schedule, and no previous work has investigated automatic adjustment of the assignment schedule from a real-time perspective.

This study aims to bridge existing research gaps by departing from a static viewpoint in gate assignments, by analyzing the gate assignment procedure as a dynamic, real-time decision-making process. Thus, our main research objective can be described as: *To develop a unified decision support system for both static and dynamic scenarios*. In other words, our model can be used at the two stages, in pre-assignment and then in real-time after disruptions occur. Our solution stands-out for being a single model that can be run at any time, with updated information,

* Corresponding author.
E-mail address: george.li@tudelft.nl (H. Li).

https://doi.org/10.1016/j.orp.2025.100338
Received 8 October 2024; Received in revised form 23 December 2024; Accepted 2 April 2025
Available online 11 April 2025

and by being able to provide a solution in a matter of milliseconds. Our objective is to explore the efficacy of this promising approach in addressing the stochastic and dynamic nature of gate allocation challenges. We propose that by leveraging the sequential decision-making of Deep Reinforcement Learning (DRL), our framework may effectively manage urgent and frequent changes. We anticipate that this approach will (1) enhance current gate allocation methods by reducing future instances of undesirable remote gate usage, (2) strike a balance between competing objectives such as minimizing passenger walking time and prioritizing flight schedules, and (3) exhibit exceptional flexibility, enabling swift execution under varying flight configurations with remarkably fast run times.

Our approach to the Gate Assignment Problem (GAP) involves two stages, each leveraging Markov Decision Processes (MDP) and DRL methodologies. MDP provides a data-based framework for automatically solving sequential decision-making problems. With MDP, the environment is modeled as a set of states and actions that can be performed to control the system's state. The goal is to control the system in such a way that some performance criteria is maximized. This modeling constitutes the base for the DRL model. DRL has the advantage of not having to rely on a complete and accurate model of the environment. Therefore, it can maximize rewards in environments where the exact dynamics of the environment are not known. We employ the Asynchronous Advantage Actor-Critic (A3C) architecture to address the GAP due to its unique properties — using multiple independent interacting agents allows for greater exploration in less time. Initially, we address the GAP from a static perspective, termed pre-assignment, where gate assignments are made before operational commencement. Subsequently, we transition to real-time assignment, where gate assignments are dynamically adjusted in response to evolving operational conditions, including real-time flight schedule changes. With each approaching flight, a decision is made, and we monitor the results before adjusting our decisions. Leveraging the inherent adaptability of MDP and RL, our framework seamlessly incorporates real-time adjustments, ensuring efficient and responsive gate allocation within airport operations.

This paper is organized as follows: Section 2 describes past GAP research as well as contemporary DRL applications in the transportation sector. In Section 3, an MDP formulation is proposed, which is tightly tailored to the GAP. Section 4 incorporates the developed MDP into the A3C architecture and builds the real-time GAP solution (REGAPS). The hypotheses for the implemented method are presented in Section 5. Section 6 conducts an empirical case study based on real airport data and evaluates and validates the model's performance. Note that the specific airport is not disclosed due to privacy issues. Section 7 discusses whether the initial hypotheses were verified. Finally, Section 8 concludes this paper.

2. Related work

This section covers the state-of-the-art on research on gate assignment. Previous works on this topic are described in Section 2.1. Additionally, given that DRL is prominent in this work, Section 2.2 describes the latest advances in this area outside of GAP. Finally, Section 2.3 will clearly define the gaps in literature to be covered by this work.

2.1. Gate Assignment Problem (GAP)

Research into GAP has a long history in the operations research discipline due to its practical importance. There are multiple types of approaches with respect to the solution methodologies. These can be categorized into four parts:

- *Expert systems* design software to simulate the decision-making process of human experts. Some studies that fall into this category are Brazile & Swigger [7], Gosling [8], Srihari & Muthukrishnan [9] and Su & Srihari [10].
- *Exact methods* are one of the most focused methodologies for GAP mainly modeled as Integer programming or Linear Programming. For instance, Barbic [11] published one of the earliest studies using a branch and bound algorithm with a lower bound that minimizes the total walking distance of the passengers. Mangoubi & Mathaisel [12] proposed a greedy heuristic approach and a linear programming relaxation method to minimize the total walking distance. Bolat [13] focused his objective on minimizing the difference between the minimum and maximum slack times, using a branch-and-bound algorithm and a heuristic approach called branch-and-trim. The primary limitation of exact methods is the extensive computation time. In practical GAPs, the objective is not to find a globally optimal solution but rather to obtain a satisfactory solution within a reasonable time frame.
- *Stochastic Processes and Heuristic*: this is the example of the work from Haghani and Chen [14] that proposed a heuristic algorithm to minimize passenger walking distance by assigning successive flights to the same slot. Ding et al. [15] developed a greedy algorithm to acquire an initial schedule for the hybrid algorithms based on Tabu search and Simulated Annealing later on. Recently, Jie Li et al. [16] presented a column generation-based algorithm to solve gate assignment with combinational gates. Jiang et al. [17] introduce a novel approach to optimizing airport gate assignments, addressing harbor apron safety constraints through a two-phase mathematical model. Refinements to the branch-and-price method improve efficiency and accuracy in solving the problem. She et al. [18] addresses the Airport Gate Assignment Problem (AGAP) by proposing a multi-objective integer programming model and a two-phase Monte Carlo-based NSGA-II algorithm. Computational analyses validate the efficacy of the approach in providing economical, robust gate assignments. The drawback of stochastic processes lies in their inherent high computational complexity, coupled with the challenge of generating realistic scenarios.
- *Reinforcement learning (RL) approaches:* these offer an advantage over stochastic processes by not needing to specify a model. RL, the subject of this paper, is model-free, providing a data-driven, learning-based framework to formulate and solve sequential decision-making problems. RL has been widely used in aviation, with applications ranging from airline revenue management to aircraft altitude control [19]. Many aviation-related scenarios can be conceptualized as sequential decision problems, making RL an ideal tool. In various domains, RL has demonstrated promising outcomes. Nevertheless, to the best of the authors' knowledge, the work Yildizi et al. [20] stands as the pioneering and sole attempt to apply RL to gate assignment. Their preliminary study uncovered a time complexity issue when dealing with more than 12 gates. Nonetheless, a variety of RL algorithms and techniques exist, and determining the most suitable one for each problem requires extensive examination. It is evident that further research is warranted to explore the potential of RL for GAP. To this end, we propose a more complex RL algorithm capable of handling a larger number of gates.

Despite all the effort that has been done to assist on-ground operations, few of them are actually implemented in airports. One of the major issues is that these approaches are mainly designed from a static perspective. These solve the Pre-assignment problem, which is considerably different from the final gate assignment once perturbations in real-time operations occur. Although some solutions are formulated as stochastic and robust approaches, and try to solve GAP from a dynamic perspective, with the limitation of heuristic algorithms, these can only minimize idle time and gate conflicts, but do not fully adjust to future uncertainties and perturbations.

Furthermore, the scale of the problem warrants the utilization of methods that are not reliant on specific models and can effectively

incorporate the temporal dimension of action sequences. First, the fact that RL learns by direct training in the environment renders it applicable to any airport configuration. Note that the decision space for this problem can vary largely, from the number of flights to the multitude of gates, depending on the unique characteristics of each airport. Second, gate assignment represents a sequential dilemma wherein the allocation of specific gates at present influences the future availability of gates for subsequent aircraft. RL has the ability to consider future states when making decisions. Lastly, for large airports, real-time evaluation using conventional methods is not a solution, given their larger execution times.

2.2. Deep Reinforcement Learning (DRL)

To find the next-generation solution for the GAP, we resort to DRL. The foundation of RL is the MDP, which is molded with the gate assignment process. This section explores DRL works employed in other areas. RL aims to achieve a maximum reward through actions made by the agent and the interaction reward with the environment. The RL framework has become promising due to its ability to learn the dynamics of the environment through direct interaction with the environment. With improved data availability and computational power, RL studies in aviation now range from airline revenue management to air traffic control (ATC) [19].

Recently, DRL has emerged. This coalesces deep learning (DL) and RL together and exploits both of their advantages the most. DL is widely used for function approximation and value prediction. There have been many precedents regarding DRL implementation in the transportation discipline. Lin [21] proposed a DRL approach toward the Electric Vehicle Routing Problem with Time windows. Yu [22] offered a solution for online vehicle routing with neural combinatorial optimization and DRL. There are also applications on smart transportation systems concerning the rebalancing problem of the Bike-sharing system [21]. Li et al. [23] present a novel heuristic algorithm for the Gate Assignment Problem (GAP) in airport management. Combining tabu search with reinforcement learning, it efficiently explores solutions, outperforming existing methods in solution quality and computation time across real-world benchmarks. Sui et al. [24] introduce a tactical conflict resolution strategy using Deep Reinforcement Learning to mitigate the increasing risk of flight conflicts in airspace. Controllers' actions are modeled as a Markov Decision Process, trained by the Deep Q Network algorithm. Simulation experiments confirm the strategy's feasibility and alignment with real-world flight safety regulations. Zhang et al. [25] address the Multi-Trip Vehicle Routing Problem with Time Windows using a novel Coordinated Multi-agent Hierarchical Deep Reinforcement Learning approach. By structuring a three-layered framework, this method enhances solution quality and convergence rates, outperforming traditional heuristic algorithms and reinforcement learning techniques. Results indicate significant improvements in cost effectiveness and operational robustness, demonstrating the effectiveness of the proposed approach in transportation scheduling.

DRL has also been employed to address dynamic resource allocation problems. In this context, gate allocation can also be conceptualized as a type of resource, suggesting that insights from this research domain may be relevant. Jia Wang et al. [26] propose an incremental reinforcement learning framework for dynamic resource allocation, where task patterns are extracted from large-scale data. They construct an environment model that enables a learning agent to infer the logic of task service operations and calculate feedback scores for each allocation decision. Applications of dynamic resource allocation also extends to the transportation sector. For instance, Ying He et al. [27] propose a general framework that enables fast-adaptive resource allocation in dynamic vehicular environments by integrating hierarchical reinforcement learning with meta-learning. This approach allows the framework to quickly adapt to new environments by fine-tuning only the top-level master network, while the low-level sub-networks continue to make optimal resource allocation decisions. In another example, Hongbin Liang et al. [28] model the resource allocation problem in the Internet of Vehicles as a semi-Markov decision process, incorporating a resource reservation strategy and a secondary resource allocation mechanism. A RL algorithm is applied to solve the model.

2.2.1. DRL algorithms

DRL is also a generalized term for a large class of algorithms. One branch of DRL is called value-based approaches. The algorithm's core is the value function that directly indicates the value or reward of an action. Deep Q-learning Network (DQN) [29] is a commonly used DRL model. However, it tends to overestimate the Q-value [30]. Many attempts have been carried out to solve this deficiency, and many variants emerged, such as Double DQN (DDQN) [31], and Dueling DQN [32]. A second branch is Policy-based approaches. The difference from value-based approaches is that rather than directly adjusting the action based on the reward, it adjusts the probability of choosing a certain action. A commonly used policy-based algorithm is the REINFORCE algorithm [33]. There are also many other researchers who have adopted DRL in the transportation field [34–38].

In recent years, actor-critic algorithms [39] have been proposed to take advantage of the best properties of value-based and policy-based algorithms. These are divided into two parts: (1) generating an action based on a state, and (2) computing the Q-value of the action. The actor takes an action based on the given state, while the critic evaluates the action with the value-based function. Then the actor adjusts the probability of the selected action based on the critic's evaluation. Even though these methods sound promising, implementing them has many difficulties. Actor-critic requires Artificial Neural Network (ANN) training both in the Actor and Critic models, which no doubt imposed great difficulty in network training. Also, the Q-value generated by the critic model is unable to tell the actor how much better or worse is the chosen action. The Asynchronous Advantage Actor-Critic (A3C) [39], used in this work, is one of the solutions to conquer these drawbacks. This model asynchronously parallel trains different agents on multiple independent environments. Each agent will update the global network asynchronously. This strategy not only exploits most of the computing capability of a workstation but also greatly accelerates the training process [40–42]. Note that the A3C algorithm was picked over an Advantage Actor–Critic (A2C) approach due to its asynchronous aspect, which has shown faster learning times [43] on a multi-core CPU computer as used by the authors.

2.3. Literature gaps

The review of the existing literature underscores a critical gap in addressing the dynamic nature of the GAP, which has predominantly been approached from a static perspective, focusing on Pre-assignment. Despite sporadic efforts to incorporate stochastic and robust methods, existing solutions remain hindered by heuristic algorithms and struggle to adapt to real-time perturbations and uncertainties. In response to this challenge, our study proposes a novel approach that integrates DRL techniques with the MDP framework. Specifically, we employ the A3C algorithm, renowned for its parallel training capability, enabling efficient decision-making in time-sensitive scenarios.

What sets our approach apart is its unique capability to seamlessly address both pre-assignment and real-time assignment challenges with minimal modification. By leveraging the synergistic capabilities of DRL and MDP, our framework offers enhanced flexibility and adaptability in optimizing gate assignments across various operational scenarios. Comparing with other approaches, DRL has the following advantages:

- The process of customizing the DRL environment and reward function is straightforward. Operators can easily design or modify gate-related information, such as the number of gates, their

distance from security checkpoints, and the specific flights assigned to each gate, based on the unique requirements of the airport. This flexibility enables airports to tailor the system to their operational needs without requiring advanced technical expertise.
- DRL allows us to anticipate future changes and disruptions while generating fast and effective solutions to support operators in making decisions under tight time constraints. Operators can easily incorporate changes and disturbances by updating the state space, enabling the system to adapt swiftly to evolving conditions and ensuring that decision-making remains responsive and efficient.
- DRL is capable of handling both static and dynamic scenarios simultaneously or switching between them as needed by operators, providing the flexibility to accommodate varying operational demands. Depending on the operator's requirements, DRL can generate either pre-assignment results or real-time assignment results, allowing the system to adapt to different scheduling and decision-making contexts efficiently.

3. Model formulation

The GAP poses a significant operational challenge in airports, revolving around the allocation of arrival and departure flights to suitable gates. The primary goal is to optimize gate assignments, taking into account key factors like minimizing passenger walking distances, maximizing gate utilization, and mitigating operational delays. In our research, we incorporate real-time gate assignment using reinforcement learning techniques to fully harness its potential advantages. Note that the real-time assignment is applied after the pre-assignment, considering the deviations between the actual and original flight schedules as the disturbance.

This study models a gate assignment problem as a Markov decision process (MDP). We consider that an airport has a series of nodes $I = \{1, 2, \dots\}$. Each node is regarded as a gate at the airport for aircraft to park temporarily and for passenger alighting and boarding. The set of gates I is classified into I_b and I_a, where I_b is the subset of nodes in I that are Bridge Gates and I_a are the Apron gates (Remote gates). A bridge gate is a gate that is directly connected to the terminal with a bridge, while a remote gate needs a shuttle bus to transfer passengers between the terminal and the aircraft. We define $F = \{1, 2, \dots\}$ as the set of flights. The optimization problem here is to ensure that every flight in F has a gate in I while minimizing the total cost.

Section 3.1 presents the key assumptions, followed by demonstrating the state transition process in the MDP (Section 3.2) and the objective functions (Section 3.4). Finally, Section 3.5 formulates the constraints imposed on the decision variables.

3.1. Key assumptions

The following assumptions are considered in this work:

Assumption 1: Similar type of flights occupy the gate with a similar amount of time, and the aircraft will be towed away or ready for departure. Aircraft will not occupy the gate just for waiting purposes. The occupation time of the gate consists of several parts, as shown in Fig. 1:

(a) Taxiing/towing time.
(b) Ground service.
(c) Passenger alighting and boarding.
(d) Idle time.

Assumption 2: We assume that the capacity of the airport apron is infinite. When there is no vacant bridge gate, the aircraft will be assigned to the Apron.

Assumption 3: Considering that the gate assignment process occurs subsequently to obtaining the flight schedule, we assume that all flights can successfully land without encountering any additional infrastructure complications. Limited runway capacity, or priorities between airlines, for example, are not considered.

3.2. State formulation

The RL algorithm has a discrete state space. Given the infrastructure and settings, the GAP is regarded as an MDP in which each stage n refers to the flights in F. Each stage n is characterized by the corresponding tuple of state variables s_n, and influenced by the decision variables x_n. Herein, we define the states tuple:

$$s_n = (f_n, \tau_n). \tag{1}$$

Each state n contains the arrival times f_n, which is determined in pre-assignment scenario and dynamically updated in real-time assignment scenario and the gate size of the required flight (medium or large), τ_n. The gate sizes considered are as follows:

- $\tau_n = 2$: medium size, can accommodate small or medium aircraft.
- $\tau_n = 3$: large size, can accommodate small, medium, or large aircraft.

Note that a larger gate can also be used by smaller aircraft. The distinction between small, medium, and large aircraft is clarified in following Section 3.3.

Finally, note that additional information could be used to hopefully provide more information on future perturbations. However, it was decided to keep this simplified state formulation for the following reasons:

- Ensuring method agnosticism with respect to airport layout allows for versatile application across various airports.
- Environmental factors, such as weather conditions, are deliberately excluded within this approach. The intention is not to predict disturbances but, instead, to react to them within the decision-making framework of the GAP at the airport.
- With this state formulation, the algorithm utilizes identical inputs to those of conventional methods employed as baselines. This alignment facilitates direct comparisons,
- Future flight schedules contain information for the complete day, which warrants a large state space. Increasing this state would further considerably increase the training time of the algorithms.

3.3. Action formulation

There is a set of discrete actions, where each one represents a gate that the agent may select. For each state, the number of possible actions corresponds to the number of available gates. Consequently, this RL model can be run for different airport configurations, with the only modification being the number of gates. Each action, x_n, is characterized as a tuple, and on top of the gate number, it has multiple attributes which represent the limitations of the gate:

$$x_n = (I_n, \lambda_n, \eta_n, \omega_n, \rho_n). \tag{2}$$

I_n represents the gate number, which is the actual element that agent chooses, other elements serve as constraints. λ_n is the compatible aircraft size for this gate:

- $\lambda_n = 1$: small aircraft (i.e. under 100 seats).
- $\lambda_n = 2$: medium aircraft (i.e. between 100 and 200 seats).
- $\lambda_n = 3$: large aircraft (i.e. more than 200 seats).

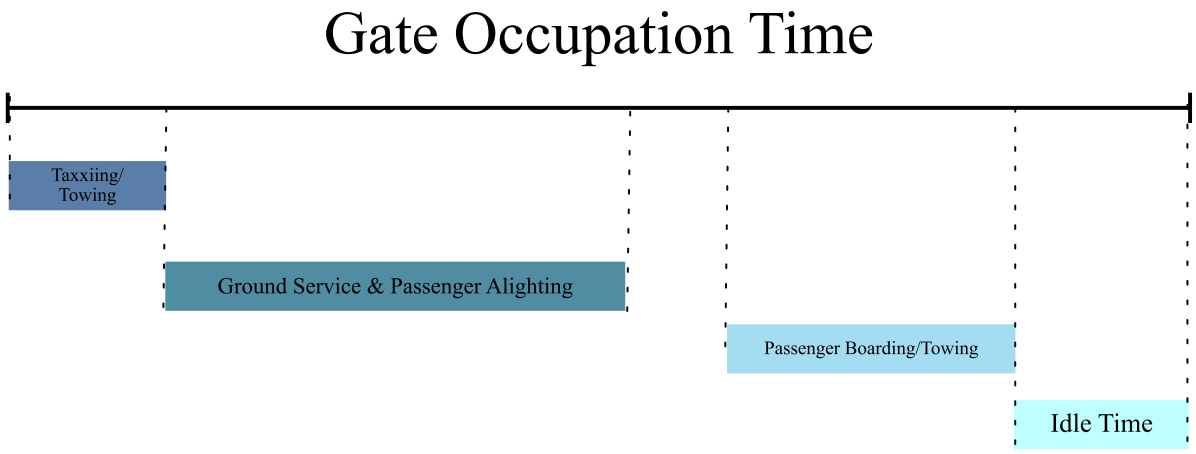


Fig. 1. The several phases of the gate assignment process.

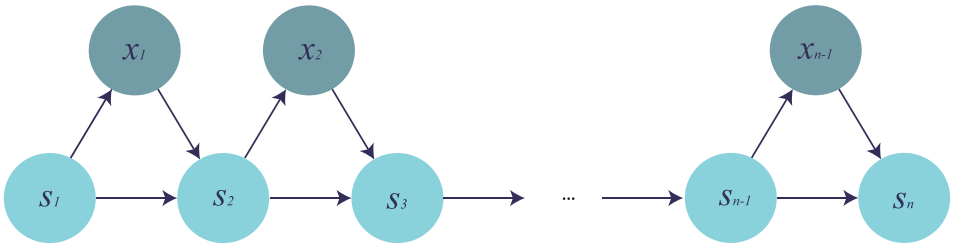


Fig. 2. State transition.

η_n is the occupation time for this gate. Finally, ω_n is the earliest available time, and it is determined as follows:

$$\omega_n = \hat{f_{I_n}} + \hat{\eta_{I_n}} \tag{3}$$

where $\hat{f_{I_n}}$ is the flight arrival time the last time Gate I_n was chosen. $\hat{\eta_{I_n}}$ is the gate occupation time when last time Gate I_n is selected. ρ_n is a binary indicator showing whether this gate is an Apron gate.

The different settings of x_n influence the overall performance of the gate assignment. It was chosen to define all possible actions in the form of x_n, instead of just the gate identification, as to give more information to the RL model to make its decision.

The terminal condition of this MDP is when all the inbound flights get their gate assignment command. Fig. 2 demonstrates the state transition process, characterized by the state transition function $p(s_n, x_n)$.

3.4. Objective formulation

Given the current circumstances and the requirements of the airport ground operations, a multi-objective approach is preferred. Based on the performance selection method [44], (1) the number of the remote gate and (2) passenger walking time are applied as the objectives in this paper. These objectives are then combined into a single reward that is given to the RL agent. The weight of two objectives are determined according to the priority lined out in [44].

The number of remote gates cost, L, is denoted as:

$$L = \sum_{n=1}^{f} \rho_n \tag{4}$$

We interpret Passenger walking time as an average man walking time taken from the security check to the boarding gate, denoted in Eq. (5).

$$t_{pwt,I_n} = \frac{d_{pwt,I_n}}{v}, \tag{5}$$

where $t_{(pwt,I_n)}$ is the passenger's walking time from the security check to the chosen Boarding Gate I_n at State n. Parameter $d_{(pwt,I_n)}$ is the distance from security check to the chosen Boarding gate I_n at State n. In turn, v is the average human walking velocity (i.e. 5.43 km/h). The objective is to minimize the total PWT, denoted in Eq. (6).

$$T_{pwt} = \sum_{n=1}^{f} t_{pwt,I_n} \tag{6}$$

With the two cost components mentioned above, the total cost function is demonstrated as follows:

$$C = \alpha L + (1 - \alpha) T_{pwt}, \tag{7}$$

where α is a coefficient that balances the two objectives' trade-offs. We assume $\alpha = 0.8$, as we come up with this number upon agreement with airport operators. The overall objective function for the GAP is then defined:

$$min[C], \tag{8}$$

3.5. Constraints

GAP is subjected to a set of regulatory constraints. Constraint (9) demonstrates the selected gate must be one of the functioning gates:

$$I_n \in I \tag{9}$$

Note that functioning means that the gate is working and can be used. Constraint (10) shows the lower (upper) bounds of the gate occupation time (η), which is different according to Assumption 1:

$$\eta_{min} \leq \eta_n \leq \eta_{max} \tag{10}$$

Constraint (11) indicates the type of the gate:

$$\rho_n = \begin{cases} 1 & \text{if } I_N \in I_a, \forall n = 1, 2, 3, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where I_a indicates Apron gates, and I_b Bridge gates. There are also a set of operational constraints that must be complied with. Constraint

(12) defines that the selected gate must be available when the flight is arriving:

$$\omega_n \leq f_n, \tag{12}$$

where ω_n represents the earliest available time of the gate, and f_n the arrival time of the flight. Finally, constraint (13) sets that the gate size, τ_n, must be compatible with the aircraft size, λ_n:

$$\lambda_n \leq \tau_n. \tag{13}$$

4. Deep reinforcement learning model

DRL is a machine learning scheme where an agent interacts with the environment over a series of time steps. At each state, the agent performs an action according to the policy. The environment then changes according to the action. A reward is given to the agent — this reward evaluates how good the environmental change was. The RL model uses this reward to evaluate how good the performed action was. The long-term reward is the accumulative immediate reward when the environment reaches its terminal state. RL aims to maximize long-term reward over episodes of the training process. In other words, RL attempts to find the actions that yield better rewards.

Section 4.1 defines the components of the developed RL model: state, actions, reward, and action shaping. Section 4.2 defines the RL algorithm employed in the work, the A3C. Finally, Section 4.3 defines how the RL model is used to produce a new real-time gate assignment based on delays of incoming aircraft.

4.1. Incorporation of components

The vital components of the RL model that are used in this paper are presented here:

- **State**: Defined in Section 3.2 as $s_n = (f_n, \tau_n)$. The size of the state space in length equals the total flights.
- **Action**: Defined in Section 3.3 as $x_n = (I_n, \lambda_n, \eta_n, \omega_n, \rho_n)$. The size of the action space equals the number of available gates.
- **Reward**: Since reinforcement learning relies on the accumulation of immediate rewards, the total cost C must be decomposed. The total cost comprises two components: the cost associated with assigning flights to remote gates and the walking time cost from security to the boarding gate. These can be interpreted as the costs corresponding to two types of actions: assigning a flight to a remote gate or to a bridge gate. To reflect this, we designed two mutually exclusive terms in the immediate reward function:

$$r_n = \alpha\rho_n + (1-\alpha)(1-\rho_n)t_{pwt,I_n} \tag{14}$$

where r_n represents immediate reward at state n, ρ_n and necessary information needed in t_{pwt,I_n} can be acquired in x_n.
In Eq. (14), only one of the two terms is applied depending on ρ_n in x_n. While traditional reinforcement learning (RL) focuses on solving a maximization problem, the Gate Assignment Problem (GAP) is framed as a minimization problem. Assigning flights to apron gates is most always undesirable; therefore, the first half of the reward is taken the opposite. A negative reward signals to the agent that this action is unfavorable. For actions involving bridge gates, which traditionally result in positive rewards, we invert the term to ensure the reward remains positive while prioritizing lower walking times. This adjustment ensures that a lower walking time corresponds to a higher reward. The final immediate reward function is defined as follows, incorporating α to regulate the relative importance of the two terms.

$$r_n = -\alpha\rho_n + (1-\alpha)(1-\rho_n)t^{-1}_{pwt,I_n} \tag{15}$$

- **Action Shaping** [45–47]: this technique is used to train the agent to choose better actions. There will be various gates that are unavailable or unsuitable to choose from at each state. The chance of selecting these faulty gates should be zero. We employ an action mask to filter out all the impossible actions at each stage, forcing the agent only to choose legal actions, considerably improving learning efficiency. The action mask is implemented through a vector:

$$\text{Mask}_a(State) = [\text{mask}_{a,1}(state), \dots, \text{mask}_{a,n}(state)], \tag{16}$$

where $\text{mask}_{a,n}(state)$ is defined as follows:

$$\text{mask}_{a,n}(state) = \begin{cases} 1 & \text{if } I_N \in I_{ava}, \forall\ n = 1,2,3,\dots \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where I_{ava} represents the subset of gates that are currently functional, available, and have a size matching the aircraft and suffice all the other constraints. Thus, the mask will only enable the RL agent to pick gates that are functional and match the size of the flight and meet other requirements.
Finally, the action mask design is adaptable and can be easily customized to different settings and cases. We may set the probabilities of invalid actions to zero by adding an extra layer before the action probability calculation in the Actor network.

4.2. Asynchronous Advantage Actor Critic (A3C)

In general, the DRL algorithm has two categories: Value-based and Policy-Based. The value-based approach is a deterministic policy, in which when the model is optimized to the best, every state's corresponding action is determined, while Policy-based is stochastic. In a policy-based algorithm, every state/action series in each episode is denoted as Trajectory (τ):

$$\tau = \{s_1, x_1, s_2, x_2, \dots, s_n, x_n\} \tag{18}$$

The probability of this trajectory is:

$$\begin{aligned} p_\theta(\tau) &= p(s_1)\pi_\theta(x_1|s_1)p(s_2|s_1,x_1)\pi_\theta(x_2|s_2)\dots \\ &= p(s_1)\Pi_{n=1}^{N}\pi_\theta(x_n|s_n)p(s_{n+1}|s_n,x_n) \end{aligned} \tag{19}$$

The total reward of this trajectory is:

$$R(\tau) = \sum_{n=1}^{N} r_n \tag{20}$$

The goal is to find the best policy π_θ, that can optimize the total reward $R(\tau)$.

However, it is difficult to find a satisfying policy. To solve this drawback, the Actor-Critic (AC) framework has been put forward. In the AC algorithm, the agent is comprised of an actor and a critic. The actor acts under the current state using the current policy, then the environment will switch to a new state and return the immediate reward to the critic. The critic will judge the action based on the reward and return it to the actor for updating and calibrating the policy. A3C is proposed based on the AC algorithm, where the difference is that A3C employs multiple agents working concurrently, and asynchronously trains the Artificial Neural Network (ANN), therefore accelerating the training process while enhancing the convergence significantly.

Here, we present the working process of one agent.

Similar to AC algorithm, The A3C algorithm has a policy function $\pi_\theta(x_n|s_n)$, and a value function $V_{\theta_v}(s_n)$,where θ, θ_v are weights parameters. Both policy and value functions are updated after certain steps of action. The value of the discounted reward, $DR(\tau)$, tells the agent which actions are rewarding and which are not.

$$DR(\tau) = r_1 + \gamma r_2 + \gamma^2 R_3 \dots = \sum_{n=1}^{\infty} \gamma^{n-1} r_n \tag{21}$$

where $\gamma(0 < \gamma < 1)$ is the discount factor. The greater discount factor the more the agent is more focused on the future reward. Thus, the value function is defined as:

$$V_{\theta_v}(S_n) = E[DR(\tau)|s = s_n, \pi] = E[\sum_{n=1}^{\infty} \gamma^{n-1} r_n | s = s_n, \pi] \tag{22}$$

Rather than use one-step Time-Difference-error(TD-error) in AC, we use n-step TD-error to better update the parameters. The update performed can be seen as:

$$\nabla_\theta J(\theta) = \nabla_\theta log\pi_\theta(x_n|s_n) A_{\theta,\theta_v}(s_n, x_n) \tag{23}$$

$$A_{\theta,\theta_v}(s_n, x_n) = Q(s_n, x_n) - V_{\theta_v}(s_n) \tag{24}$$

where $Q(s_n, x_n) = \sum_{n=1}^{N} \gamma^{n-1} r_n + \gamma^n V_{\theta_v}(s_{n+1})$.

To encourage exploration and avoid premature convergence, an entropy item is adopted. The Actor network parameter update is performed based on the following accumulated gradient:

$$d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi_{\theta'}(x_n|s_n) A_{\theta',\theta_v}(s_n, x_n) + \beta \nabla_{\theta'} H(\pi_{\theta'}(s_n)) \tag{25}$$

where $H(\pi_{\theta'}(s_n))$ is the entropy, β controls the relative importance of the entropy and the reward. The Critic-network parameter update is performed based on the following accumulated gradient:

$$d\theta_v \leftarrow d\theta_v + \frac{\partial (A_{\theta',\theta_v}(s_n, x_n))^2}{\partial \theta'_v} \tag{26}$$

Finally, the proposed A3C-based Gate assignment algorithm is demonstrated in Algorithm 1 [29].

Algorithm 1 A3C-based Gate Assignment Algorithm

Initialization:
Initialize global actor-network and critic network parameters with θ and θ_v
Set global Episode Counter T=0 and thread episode counter t=0
Initialize thread actor-network and critic network parameters with θ' and θ'_v
Initialize $T_{max}, t_{max}, N, \gamma, \beta$
DO:
Set t = 1
Reset $d\theta = 0$ and $d\theta_v = 0$
while $T < T_{max}$ **do**
    Retrieve parameters from the global network: $\theta' = \theta, \theta'_v = \theta_v$
    Obtain state s_n
    while $t < t_{max}$ **do**
        Choose action x_n under the policy $\pi_\theta(x_n|s_n)$
        Get reward r_n and new state s_{n+1}
        $t = t + 1$
    end while

$$Q = \begin{cases} 0, terminal\,state \\ V_{\theta'}(s_n), non - terminal\,state \end{cases} \tag{27}$$

    while $t = t_{max}$ **do**
        $Q = r_n + \gamma Q$
        Update thread actor-network parameter:

$$d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi_{\theta'}(x_n|s_n) A_{\theta',\theta_v}(s_n, x_n) + \beta \nabla_{\theta'} H(\pi_{\theta'}(s_n)) \tag{28}$$

        Update thread critic-network parameter:

$$d\theta_v \leftarrow d\theta_v + \frac{\partial (A_{\theta',\theta_v}(s_n, x_n))^2}{\partial \theta'_v} \tag{29}$$

    end while
    Perform asynchronous update θ and θ_v using $d\theta$ and $d\theta_v$
    T= T+1
end while

The global framework of the A3C algorithm is illustrated in Fig. 3. The agent is comprised of an actor and a critic. The actor acts under the current state using the current policy. Then the environment will switch to a new state and return the immediate reward to the critic. The critic will judge the action based on the reward and return it to the actor for updating and calibrating the policy. A3C is proposed based on the Actor-Critic algorithm, where the difference is that A3C employs multiple agents working concurrently and asynchronously training the ANN, accelerating the training process while enhancing the convergence significantly.

4.3. Real-time Gate Assignment Problem Solution (REGAPS)

With the successful implementation of the A3C-GAP algorithm, the Gate Pre-assignment Problem has been solved. However, the pre-assignment schedule always deviates from the actual schedule due to multiple causes, such as weather conditions and regulations. Hence, the proposed REGAPS algorithm diagram is demonstrated in Fig. 4. At Time Zero, which marks the beginning of the day, the model initiates the execution of an algorithm that treats gate assignment as a pre-assignment problem, relying on the original flight schedule. Under normal circumstances, the operator proceeds with gate assignments per the predetermined schedule when no delays are encountered. However, upon detecting a delay (at Reschedule point T, Fig. 4), the model promptly updates the flight schedule and re-executes the algorithm using the revised input, thus obtaining an updated assignment. The figure labeled as Fig. 5 provides a detailed explanation of how flight schedules are updated. Note that in this paper, we assume that once the gate occupation time (which includes boarding time) is completed, the aircraft will either be ready for takeoff or moved to the apron in the event of a delay, as Assumption 1. Therefore, departure delays do not affect gate usage in our current model. It shows that when delays or changes in schedules are identified, only the arrival times of affected flights are adjusted, while the assigned gates and arrived flights remain the same. This iterative process ensures the attainment of real-time gate assignments, facilitating their completion in response to changing conditions.

To solve the real-time assignment problem, the initialization should be set as the new timetable of the arriving flights, and the gate occupation states should be set simultaneously. Also, several buffers are applied to store the best action, best reward, state information, and gate occupation status.

5. Hypotheses

The following hypotheses are established regarding how the REGAPS algorithm can improve gate assignment over conventional methods:

Hypothesis 1: The REGAPS algorithm is poised to surpass other methods by strategically considering the sequence of actions over time. Note that the REGAPS algorithm does not possess additional information compared to other methods, and does not predict disturbances. However, we hypothesize that its unique capacity lies in learning from past experiences. The REGAPS algorithm can discern which decisions result in more favorable outcomes. For instance, it can learn which gate allocation patterns lead to reduced Apron usage by day's end.

Hypothesis 2: The REGAPS algorithm will prioritize reducing Apron gate usage, as this value was made paramount in the reward formulation (Section 3.4).

Hypothesis 3: The REGAPS will improve upon conventional method in environments with higher complexity and uncertainties. In environments with lower complexity, it is expected that conventional methods already perform well and the introduction of REGAPS may be not represent and improvement.

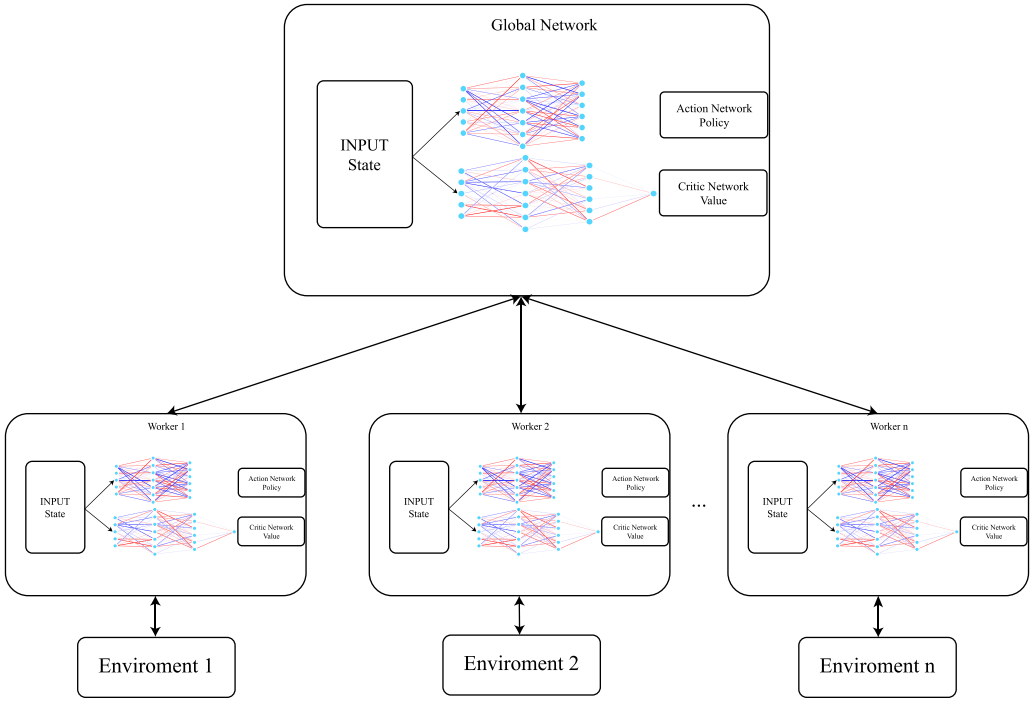


Fig. 3. The A3C framework.

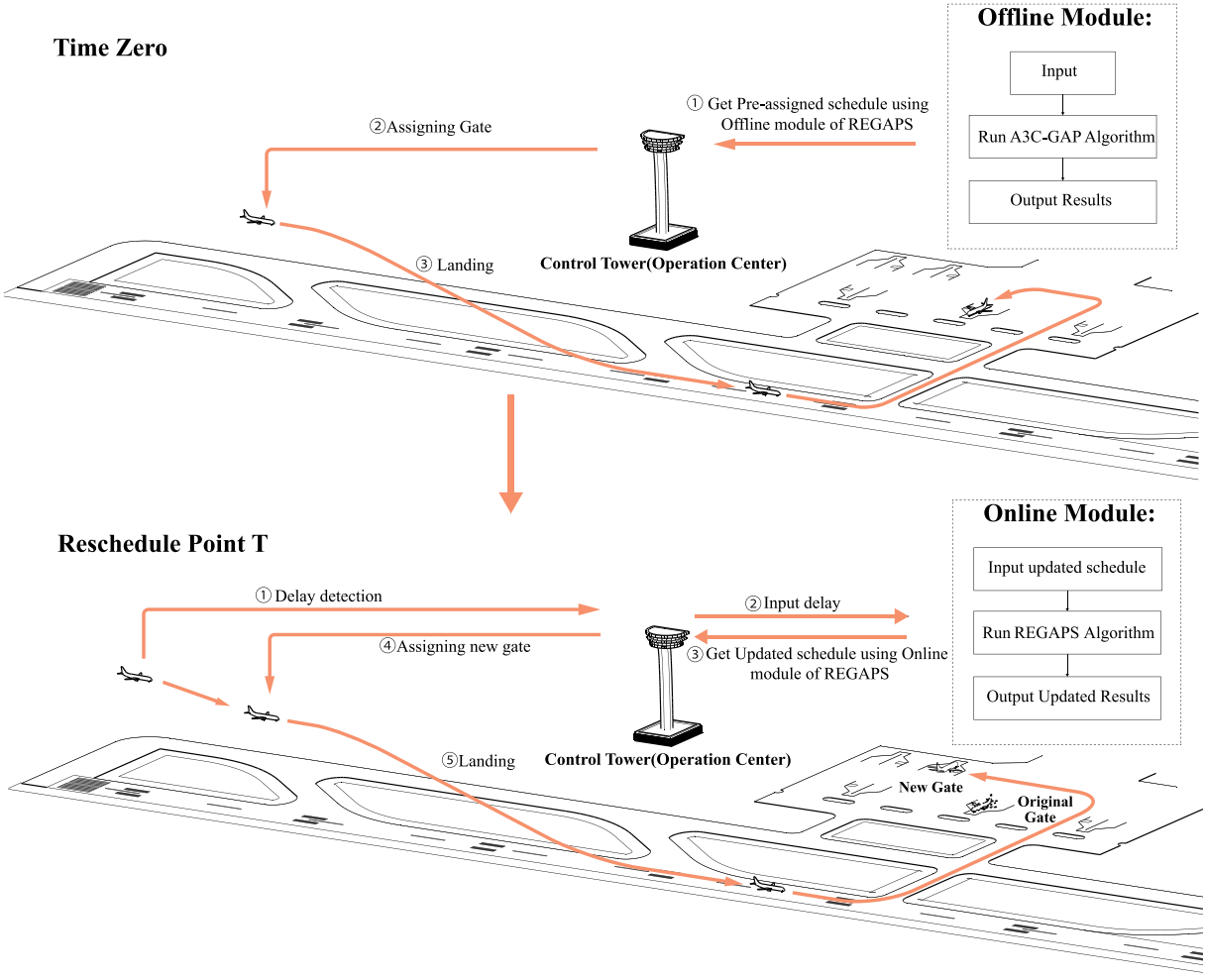


Fig. 4. Diagram of the proposed REGAPS.

Table 1
Pre-assignment experiment results.

Gate number	Quantity	Category
59,61,63,65,67,69,71,73,75,77,79,81,83,85,87,80,82,84	18	High-risk
50,51,53,54,55,56,57,58,60,62,64,86,88,90,91,92,94,95,96,97,98	21	Low-risk

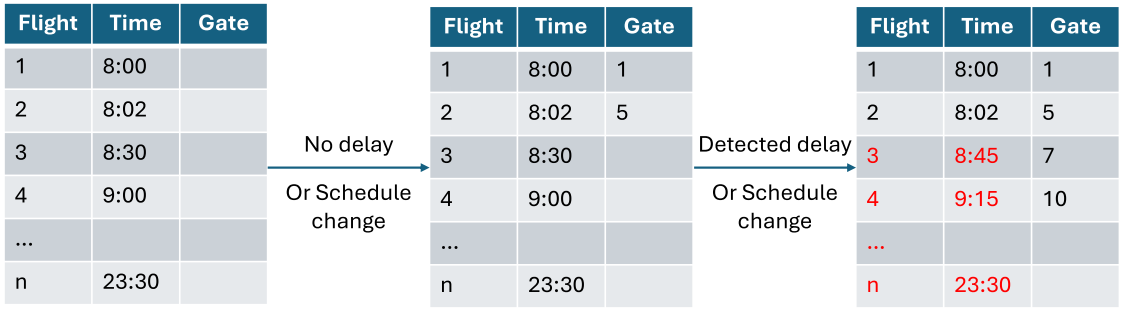


Fig. 5. Flight schedule updating diagram.

Hypothesis 4: The REGAPS algorithm is expected to exhibit superior execution speed, as RL methods are known for being more efficient speed-wise when compared to conventional methods [48].

Confirmation of these hypotheses through the performed use case will be discussed in Section 7.

6. Results

The presented results are divided into the following sections. Section 6.1 describes the settings of the conducted experiments. The methodology is first tested using actual airport data under the Non-sharing policy during COVID-19 times to verify and validate the model. Sections 6.2 and 6.3 present the pre-assignment and the real-time gate assignment results, respectively. Note that, for pre-assignment, we refer to the algorithm as A3C-GAP. Only, during the real time assignment, the term REGAPS is used. Section 6.4 applies the methodology under various assignment policies (i.e. Complete Sharing, Non-Sharing, Partial Sharing policy) to further test the model's universality. Finally, Section 7 discusses future improvements.

6.1. Experimental settings

We use real data from an airport under the COVID-19 regulations. The data covered the period from July 7th, 2021 to July 8th, 2021, containing 124 flights. The experiment was carried out on a Mac mini, M1 chip, 8 cores and 16 GB RAM. In A3C algorithm, Actor network uses following full connected network: $N \times 64 \times 128 \times A$, N is the dimension of the states, and A is the total of Actions. Critic network uses similar structure: $N \times 64 \times 128 \times 1$.

Due to the requirements of the Technical Guidelines for Airport Epidemic Prevention and Control posted by the Civil Aviation Administration of China [49], airports need to categorize arriving flights based on the risk level. And different category flights have different dedicated gates. High-risk flights (mostly international, 15 flights) and Low-risk flights (mostly domestic ones, 109 flights) are strictly separated. Based on the consultation of the on-ground operators, we assume that low-risk flights occupy slots for 60 min. Because of the strict border control and security check requirements, a high-risk flight can occupy a slot for up to 4 h (240 min). These values were provided by the operators at the target airport and, according to their statements, are similar to values applied all through China.

The layout of the airport terminal is shown in Fig. 6. The second floor (2F) is for passengers arriving from domestic and international flights. The Middle floor (MF) is for domestic flight departure, and the third floor (3F) is for security checks and international flight departure. Passengers must first arrive at 3F to complete the check-in procedure and pass the security check. Then, domestic flight passengers will go to the MF for boarding, while international flight passengers will take an independent passage to the boarding gate. Different types of passengers will take different routes inside the terminal, resulting in different walking times. In this case, domestic passengers will walk from the security check to the predestined boarding gate.

Table 2
Hyper parameters of machine learning algorithms used in this experiment.

Hyper parameter	A3C	DQN	DDQN
Learning rate	1e−6	1e−6	1e−6
Discount factor	0.96	0.96	0.96
Network update frequency	5	500	500
Hidden layers	3	3	3
Episodes	12 000	12 000	12 000
Exploration rate	–	0.3	0.3
Memory capacity	–	10 000	10 000

The number of boarding gates in Fig. 6 corresponds to the gate numbers in Fig. 7, which connects passengers and aircraft. The airport classifies its bridge gates as High-risk and Low-risk, as displayed in different colors in Fig. 7, and explicitly stated in Table 1.

The original flight schedule, actual flight schedule, and on-ground allocation results are based on the First-come-first-served (FCFS) rule and are acquired at the same airport. In this rule, gates are assigned based on the arrival times of the aircraft.

6.2. Pre-assignment experimental results and analysis

This section will conduct an experiment and comparison on the pre-assignment problem. Note that, for pre-assignment, algorithms produce a gate assignment plan for the complete day in one run. One output of all algorithms contains the gate assigned for very flight. Herein, no real-time assignment is yet performed. The A3C-GAP algorithm will perform based on the original flight schedule. Classic RL algorithms Deep Q-Learning(DQN) [29] and Double-DQN(DDQN) [31] are applied as a comparison. The hyper parameters are provided in Table 2. The number of episodes is determined to provide sufficient time for DQN and DDQN algorithms to learn and converge. Most other hyper-parameters and implementation are coming from [50].

The Genetic Algorithm (GA) is a heuristic approach, also used for comparison purposes, the detailed implementation can be viewed at [51]The objective of this GA is to minimize costs regarding less apron gates usage and less passenger walking time. The selection of a GA over

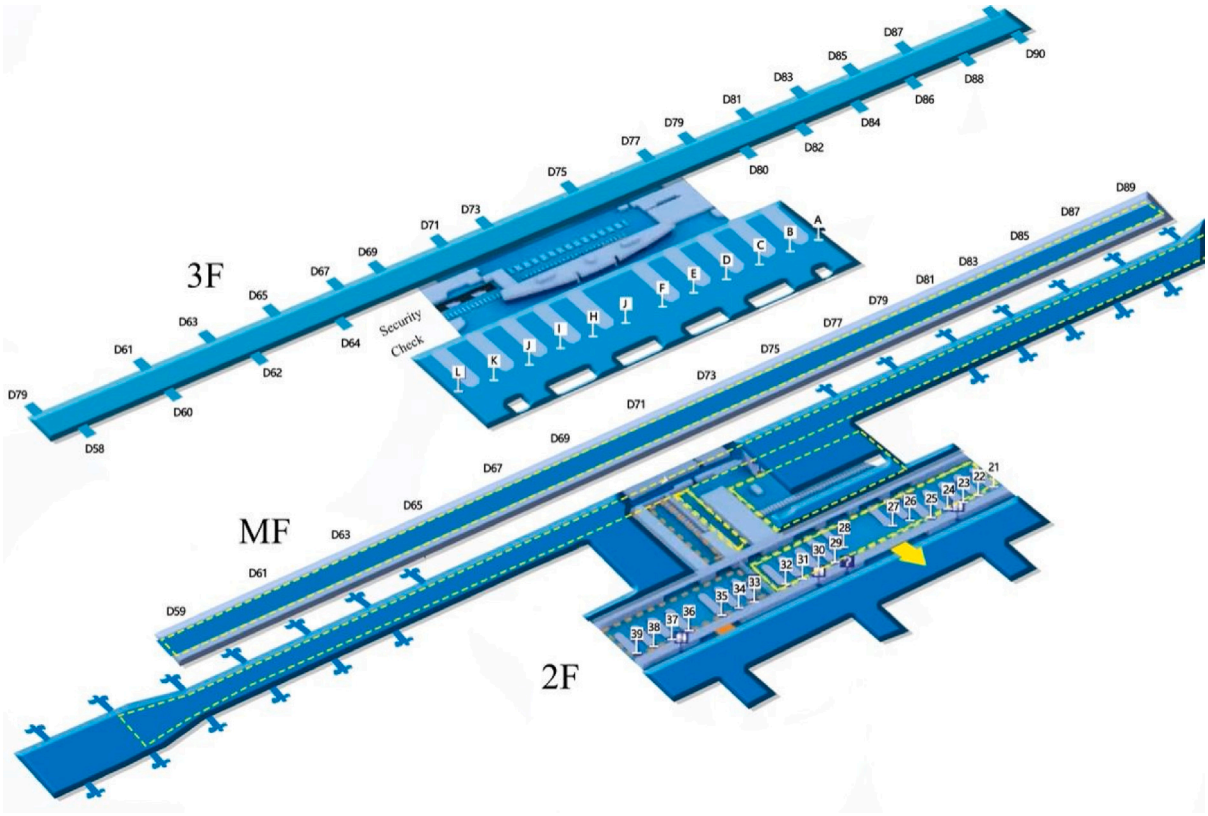


Fig. 6. Layout of the case airport.

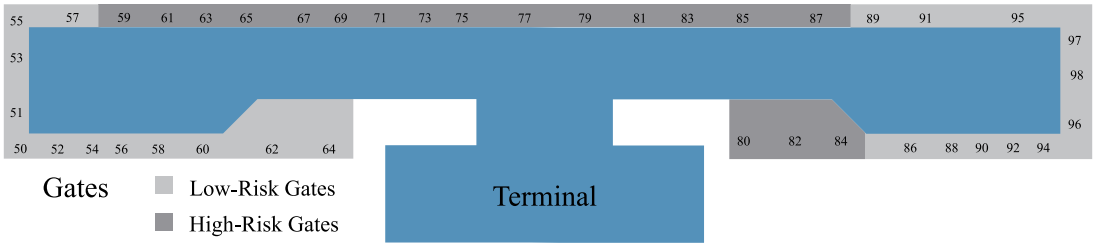


Fig. 7. Gate layout of the terminal. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

alternative methodologies was predicated upon its extensive adoption as evidenced in scholarly works [18,52,53]. The continued utilization of GA by researchers until recent times [53] underscores its enduring relevance and scientific merit, affirming its continued vitality within the field.

As previously explained, the GAP is divided into two parts: Low-risk flight assignment and High-risk flight assignment. Thus, the experiment is bipartite. All methods are run separately for these two assignments. The results of both assignments are shown side by side for comparison.

Fig. 8 displays the learning curves of each algorithm under different scenarios. We consider the best reward and assignment plan acquired from each algorithm experiment as the finalized assignment plan. As shown in Fig. 8, the proposed A3C-GAP algorithm achieves the best results in both cases. Additionally, based on the shadow area, the A3C-GAP algorithm has a much smaller variance than DQN and DDQN, which means it is more stable. According to the algorithm execution durations outlined in Table 3, A3C exhibits significantly swifter performance compared to DQN and DDQN across both scenarios, completing with a total runtime of 185 s. In contrast, both DQN and DDQN require over 1200 s to complete their execution.

Per Table 3, the A3C-GAP algorithm's best-accumulated reward is 17.42, overtaking GA's best-accumulated reward −14.67. Classic reinforcement learning algorithms DQN and DDQN acquired fairly low rewards: −44.96 and −60.56. In the realm of computational efficiency, the A3C-GAP algorithm demonstrates superior convergence speed compared to both DQN and DDQN algorithms, which exhibit prolonged convergence times and insufficient reward accumulation. This is explained by the fact due to the additional workers, A3C collects data faster. Additionally, as every single worker instance also has their own environment, the collected data is more diverse, which leads to more robust results.

The Genetic Algorithm (GA) does not lag behind the A3C-GAP algorithm. Notably, GA and A3C-GAP yield comparable outcomes in high-risk scenarios, characterized by a reduced number of flights and thereby diminished complexity. Nevertheless, the A3C-GAP algorithm stands out in terms of the performance speed. Once the complexity of the environment increases, as it is the case from high-risk to low-risk flights, the running of time GA method increases by roughly 53%. In turn, A3C-GAP increases only by roughly 19%. Regarding low-risk flights, A3C-GAP is faster than GA by almost 39%. This runtime is particularly crucial in busy airports, where prompt decision-making is imperative.

Finally, it is interesting to note that, despite the primary optimization objective was centered on Apron gates reduction, the A3C-GAP

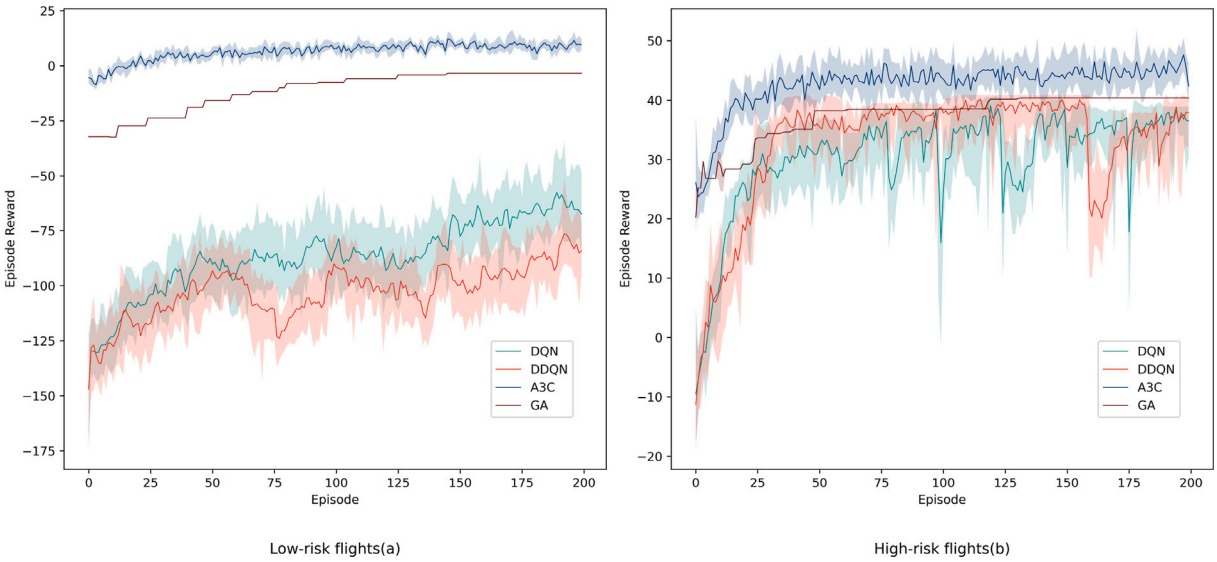


Fig. 8. Pre-assignment experiment learning curves.

Table 3
Pre-assignment experiment results.

Type of flights	Low-risk flights				High-risk flights			
Type of algorithm	REGAPS	GA	DQN	DDQN	REGAPS	GA	DQN	DDQN
Total reward [–]	17.42	−14.67	−44.96	−60.56	45	39.32	32.32	37.16
Passenger walking time [m]	1574.78	1668.89	1617.81	1640.04	33.82	48.99	51.05	30.84
Apron [–]	14	32	45	51	0	0	4	7
Algorithm running time [s]	185	302	1260	1298	156	198	653	681

algorithm effectively mitigates passenger walking time, rendering it superior across all metrics relative to alternative methods. Naturally, this is due to the fact that Apron gates are the ones with the highest walking time.

In Table 4, a segment of the assignment results is presented to better elucidate the reasons behind the superior performance of the A3C-GAP algorithm over the GA, to the extent possible due to the black-box nature of a DRL algorithm. Here, it is important to note that both algorithms have access to the same information, and optimize the same cost function. Both algorithms output a gate assignment solution for all gates during the day. Table 4 shows an excerpt of only the initial 12 flights of the day. All gates are available at the beginning of the day. It is clear that both algorithms do not select the same gates. Additionally, the GA utilized 2 apron gates — selection is made over leaving the remaining non-apron gates available for future flights. The A3C algorithm did not use any apron gate. Finally, the A3C algorithm is able to select gate '92' for two subsequent flights. This shows the higher capability of the A3C algorithm to understand the sequential nature of the GAP, resulting in reduced usage of apron gates.

6.3. Real-time experimental results and analysis

A real-time gate assignment experiment was conducted to validate the REGAPS algorithm. The algorithm for the same complete work day as the pre-assignment experiment. We consider the deviations between the actual and original flight schedules as the disturbance. Thus, the goal for the implementation of the RL model is to adapt to flight delays. Every time a deviation longer than 20 min is considered a disturbance (similarly to on-ground staff), which will trigger the reschedule module and update the flight schedule. The final comparison is between the actual assignment (i.e. as the Real case) and REGAPS.

Per Table 5, the REGAPS algorithm is significantly more effective under both scenarios than in the Real case. The REGAPS reduced both the usage of Apron gates and the total walking time. Additionally, to be noted that the low-risk flight assignment task includes a large amount of computation, whereas the high-risk task has few computation scenarios. This experiment proved that the REGAPS algorithm can yield a satisfying result under different levels of computational workload.

The respective Sankey chart of REGAPS and Real case are shown in Figs. 9(a) and 9(b). A Sankey chart is a type of flow diagram used to visualize the movement of quantities between different categories. In this context, it illustrates the distribution between apron gate and bridge gate assignments, highlighting the quantity of gates used and their respective utilization. Not only REGAPS greatly minimized the Apron gate assignment, but it also deployed fewer gates of the airport. Further, the assignment schedule is planned out evenly to avoid the intensive use of one gate. This means that REGAPS can make better use of each gate. The comparison between REGAPS and the Real case revealed that the former could achieve a significantly better outcome and reduce the occupation of airport resources.

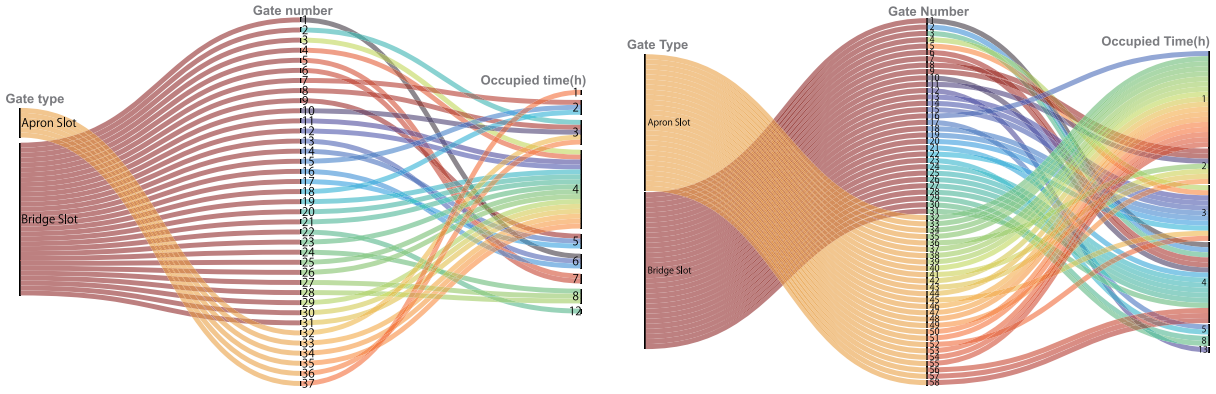
6.4. Effectiveness with policy changes

Sections 6.2 and 6.3 proved the REGAPS algorithms' validity using real-world data during the COVID period. Past COVID, the rules for gate assignments have changed. Various airports have different approaches to gate assignment. As a result, they design their policies.

According to CAAC guidelines, the optimum policy is complete gate sharing between international and domestic aircraft to maximize the use of bridge gates. However, unlike domestic passengers, international travelers must go through customs and passport checks. To allow the gate to be shared by two types of planes, airports must use a specific bridge to segregate the passenger flow line, which requires terminal and bridge modifications and construction and is costly. As a result, rather than implementing a complete sharing or non-sharing policy, airports

Table 4
Assignment examples — different actions between A3C and GA for the same scenario.

	A3C			GA		
	Gate number	Boarding distance (m)	Apron (Y/N)	Gate number	Boarding distance (m)	Apron (Y/N)
	92	1169	N	64	784	N
	56	1055	N	98	1241	N
	90	1105	N	62	830	N
	96	1232	N	57	1100	N
	57	1100	N	94	1207	N
	58	997	N	–	1241	Y
	62	830	N	56	1055	N
	55	1138	N	50	1122	N
	97	1232	N	88	1035	N
	51	1121	N	–	1241	Y
	92	1169	N	60	900	N
	86	989	N	51	1121	N
Total walking distance (m)	–	13 137	–	–	12 877	–
Passenger walking time (min)	–	145.14	–	–	142.28	–



(a) Results of REGAPS (b) Results of Real Case

Fig. 9. The Sankey chart of real-time assignment.

Table 5
Results of the real-time assignment.

Type of flights	Low-risk flights		High-risk flights	
Type of algorithm	REGAPS	Real	REGAPS	Real
Total reward [–]	12.28	−13.78	45.17	10.42
Passenger walking time [m]	1362.68	1527.1	48.25	74.72
Apron [–]	19	32	0	3

typically implement a partial sharing policy, meaning that only parts of the gates can be shared between different types of flights.

We conduct real-time gate assignment experiments with various policies to investigate the model's universality. The policies are expressed below:

Policy 1: Complete Sharing: All terminal gates are available for international and domestic aircraft.
Policy 2: Non-sharing: Gates are grouped into two sorts. Each type of gate is earmarked for exclusive use (see Section 6.1).
Policy 3: Partial Sharing: As demonstrated in Fig. 10 and Table 6. A portion of the gates are reserved for each type of flight. The remaining gates are shared.

Table 7 displays the results. Note that real data is only available for the Non-sharing policy, as only this policy was practiced during the period airport was monitored. To simulate an on-ground assignment, we apply the FCFS rule, which outperforms the real case instance. This is because there are more elements to consider during on-ground operations, such as the airline's preference or using a specific gate. FCFS can still provide a reference value at a certain level because it is better than the Real situation. REGAPS outperforms FCFS in all three policies. The passenger walking time is much shorter, and FCFS, especially under the complete sharing policy, gives the model more flexibility to explore. Also, the gates with the shortest distance are often given to international flights, as now, due to the complete sharing policy, domestic flights, which are much larger in quantity, can also use them. Furthermore, under both complete and partial sharing policies, REGAPS reduced Apron assignment to zero, which airports management strives for. These findings revealed that the REGAPS algorithm has good, and a high degree of adaptability to varied settings and requirements.

7. Discussion & Future work

This section will further discuss the results of our work at a higher level. First, taking into account the results in Section 6, the following can be said regarding the hypotheses previously elaborated in Section 5:

Hypothesis 1: The REGAPS algorithm outperforms other methods, even though it receives the same input and optimizes the same objective function. To be noted that the REGAPS algorithm is agnostic regarding the causes of disturbances, it simply reacts to future flight plans. However, the training over data of airport allows the DRL algorithm to find optimal sequences of gate allocation that result in lower Apron usage at the end of a run. Results from Table 6 show that the REGAPS algorithm uses fewer apron gates and has a better balance of gate usage. This hypothesis is confirmed. We further hypothesize that the improved efficiency of the A3C-GAP algorithm is due to the capability of RL of optimizing problems requiring sequential decision making.

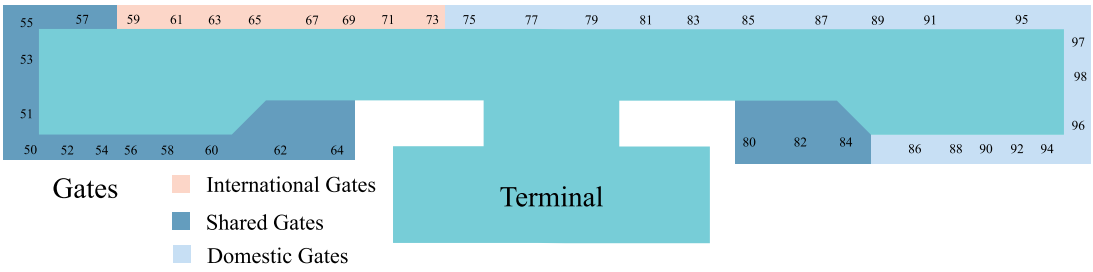


Fig. 10. Gate layout under Partial Sharing policy.

Table 6
Classification of the gates (Partial sharing).

Gate number	Quantity	Category
59,61,63,65,67,69,71,73,	8	International
75,77,79,81,83,85,87,80,82,84, 50,51,53,54,55,56,57,58,60,62,64	21	Shared
86,88,90,91,92,94,95,96,97,98	10	Domestic

Table 7
Experiment results under different policies.

Policy	Complete sharing			Non-sharing			Partial sharing		
Algorithm	REGAPS	FCFS	Real	REGAPS	FCFS	Real	REGAPS	FCFS	Real
Passenger walking time [m]	448.8	849.74	–	1410.93	1375.32	1601.88	981.34	993.21	–
Apron [–]	0	2	–	19	25	35	0	3	–

Hypothesis 2: The REGAPS algorithm had a more considerable improvement on the reduction of Apron gates (see Table 3). This hypothesis is confirmed.
Hypothesis 3: Given the results of Table 3, the REGAPS algorithm is particular relevant when the complexity of the environment increases, i.e. higher traffic density, lower gate availability. As a result, the REGAPS is not relevant in a simplest environment, i.e. a not-busy airport. This hypothesis is confirmed.
Hypothesis 4: The REGAPS algorithm was found to run faster than other algorithms (see Table 3). This hypothesis is confirmed.

In resume, experimental testing has shown that the REGAPS model has the potential to improve upon current methods for gate allocation due to its short execution time and ability to find optimal gate allocation based on multi objectives. In pre-assignment scenarios, our algorithm reduced apron gate usage by more than 50% compared to GA, DQN, and DDQN algorithms, while also decreasing the average passenger walking time. Additionally, it achieved a 40% reduction in computation time compared to the second-best algorithm, GA. In real-time assignment, REGAPS reduced apron gate usage by over 50% and significantly enhanced the total reward. It also optimized gate utilization, preventing the overuse of any single bridge gate without increasing apron gate usage, leading to a more balanced allocation of resources.

Additionally, the following remarks can be made regarding the robustness of the model. First, since REGAPS can initiate training from any intermediate state and dynamically adjust the state and action spaces, it has the flexibility to account for the impacts of various disruptions. These include handling changes in flight schedules due to diverse factors, as well as gate malfunctions, allowing the system to adapt and respond to real-time operational challenges effectively. Second, typically, airport operators receive accurate flight schedules only one to three days in advance. Our experiment, which used a two-day schedule, is representative of the workload encountered at most airports. While changes in flight and/or gate quantity can affect the training duration, these variations are unlikely to cause significant differences in the overall process. Third, REGAPS employs a stochastic policy to generate results, meaning that each run may lead to different actions while yielding similar rewards. However, this variability only impacts states that have been retrained. As long as the system delivers results before the aircraft's landing, this variation is unlikely to cause significant operational issues, ensuring that any differences in decision-making remain manageable and do not disrupt overall airport operations.

Finally, based on the model and data limits, the data we have is not a perfect representation of the current state of operations, as it used data from the COVID period. Nevertheless, using this data enable us to increase the size of the data available for training, which, we believe, ultimately improved the optimality of the method. The shown results show that the inclusion of this data ended up being favorable for training of the model. Thus, there are a few future directions to investigate. First, additional real-time data should be integrated. For example, live flight information and passenger data, or global information such as potential future flight schedules and aircraft data would enable more accurate and dynamic gate assignments, considering the latest information and minimizing disruptions. Additionally, this work should be further validated in other real-world airport environments. However, this would require collaboration with airport authorities to conduct extensive field tests to assess its performance, usability, and impact on operational efficiency.

Additionally, in the future, departure delays should be considered. This is a source of uncertainty in the model, as some gates may not be free at the expected time. This will likely require some extra information being add to state information, more specifically the estimated time a gate will become available. The MDP would then need to learn how to manage this uncertainty in gate availability.

Finally, infrastructure limitations affecting the availability of passenger bridges at each gate should be included as constraints. This should be modeled through action masks, to better reflect real-life scenarios. However, the result is the further expansion of the state and action spaces. These future works aim to enhance the REGAPS framework, expand its capabilities, and ensure its practical applicability in real-world airport operations.

8. Conclusions

A novel strategy for optimizing gate assignment is proposed in this research. To handle both the pre-assignment and real-time assignment problems, a new approach is built on the A3C RL framework

to automate the gate assignment process to achieve real-time assignment called Real-time Gate Assignment Problem Solution (REGAPS). The methodology's effectiveness is proven using real data in static pre-assignment and real-time assignment scenarios. We compared our model to two classic RL algorithms (i.e. DQN, DDQN) and a traditional heuristic algorithm (i.e. Generic Algorithm). The results reveal that REGAPS outperforms competing techniques in important performance indicators (i.e. Passenger walking time, Apron gate assignment). Additionally, a supplement experiment was run to examine the performance of REGAPS under various policies (i.e. complete, partial, and non-sharing of the terminal gates). The finding demonstrates that REGAPS outperforms other techniques under covid-19 restrictions and is also beneficial under the other mentioned policies.

Finally, future work will focus on integrating real-time data and conducting real-world implementation and validation to enhance the REGAPS framework's effectiveness in optimizing gate assignments. This method shall be applied to additional airports with a different topology of gates.

CRediT authorship contribution statement

Haonan Li: Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Xu Wu:** Writing – review & editing, Validation, Supervision, Methodology, Investigation. **Marta Ribeiro:** Writing – review & editing. **Bruno Santos:** Writing – review & editing. **Pan Zheng:** Writing – review & editing, Supervision, Data curation.

Declaration of competing interest

The authors have declared no conflict of interest.

Data availability

The data that has been used is confidential.

References

[1] Airports Council International, International Air Transport Association, Worldwide Airport Coordinators Group. Worldwide airport slot guidelines (WASG). 2022.

[2] Daş GS, Gzara F, Stützle T. A review on airport gate assignment problems: Single versus multi objective approaches. Omega 2020;92:102146.

[3] Yan S, Tang C-H. A heuristic approach for airport gate assignments for stochastic flight delays. European J Oper Res 2007;180(2):547–67.

[4] Cheng Y. A knowledge-based airport gate assignment system integrated with mathematical programming. Comput Ind Eng 1997;32(4):837–52.

[5] Xing Z, Zhang Q, Chen Z, Luo Q. Dynamic gate assignment algorithm based on attentional convolution network. In: Deng Q, editor. International conference on advanced manufacturing technology and manufacturing systems (ICAMTMS 2022). Vol. 12309, SPIE, International Society for Optics and Photonics; 2022, p. 123092V.

[6] Xing Z, Wang S, Qiao D, Luo Q, Cheng H, Wen T. Dynamic allocation method for airport gates based on capacity relaxation. In: 2020 7th international conference on information science and control engineering. ICISCE, 2020, p. 522–6.

[7] Brazile R, Swigger K. GATES: an airline gate assignment and tracking expert system. IEEE Expert 1988;3(2):33–9.

[8] Gosling GD. Design of an expert system for aircraft gate assignment. Transp Res A: Gen 1990;24(1):59–69.

[9] Srihari K, Muthukrishnan R. An expert system methodology for aircraft-gate assignment. Comput Ind Eng 1991;21(1–4):101–5.

[10] Su YY, Srihari K. A knowledge based aircraft-gate assignment advisor. Comput Ind Eng 1993;25(1):123–6.

[11] Babić O, Teodorović D, Tošić V. Aircraft stand assignment to minimize walking. J Transp Eng 1984;110(1):55–66.

[12] Mangoubi RS, Mathaisel DFX. Optimizing gate assignments at airport terminals. Transp Sci 1985;19(2):173–88.

[13] Bolat A. Assigning arriving flights at an airport to the available gates. J Oper Res Soc 1999;50(1):23–34.

[14] Haghani A, Chen M-C. Optimizing gate assignments at airport terminals. Transp Res A: Policy Pr 1998;32(6):437–54.

[15] Ding H, Lim A, Rodrigues B, Zhu Y. The over-constrained airport gate assignment problem. Comput Oper Res 2005;32(7):1867–80.

[16] Li J, Li K, Tian Q, Jin X. A column generation-based algorithm for gate assignment problem with combinational gates. Expert Syst Appl 2024;238:121792.

[17] Jiang Y, Wang Y, Hu Z, Xue Q, Yu B. Airport gate assignment problem with harbor constraints based on Branch-and-Price algorithm. Transp Res E: Logist Transp Rev 2023;176:103192.

[18] She Y, Zhao Q, Guo R, Yu X. A robust strategy to address the airport gate assignment problem considering operators' preferences. Comput Ind Eng 2022;168:108100.

[19] Razzaghi P, Tabrizian A, Guo W, Chen S, Taye A, Thompson E, Bregeon A, Baheri A, Wei P. A survey on reinforcement learning in aviation applications. 2022.

[20] Muhafız Yıldız M, Avcı U, Örnek MA, Öztürk C. Flight gate assignment problem with reinforcement learning. In: Kahraman C, Sari IU, Oztaysi B, Cebi S, Cevik Onar S, Tolga Ac, editors. Intelligent and fuzzy systems. Cham: Springer Nature Switzerland; 2023, p. 189–96.

[21] Lin B, Ghaddar B, Nathwani J. Electric vehicle routing with charging/discharging under time-variant electricity prices. Transp Res C: Emerg Technol 2021;130:103285.

[22] Yu JJQ, Yu W, Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans Intell Transp Syst 2019;20(10):3806–17.

[23] Li M, Hao J-K, Wu Q. Learning-driven feasible and infeasible tabu search for airport gate assignment. European J Oper Res 2022;302(1):172–86.

[24] Sui D, Ma C, Dong J. Conflict resolution strategy based on deep reinforcement learning for air traffic management. Aviation 2023;27(3):177–86, Cited by: 0; All Open Access, Gold Open Access.

[25] Zhang Z, Qi G, Guan W. Coordinated multi-agent hierarchical deep reinforcement learning to solve multi-trip vehicle routing problems with soft time windows. IET Intell Transp Syst 2023;17(10):2034–51.

[26] Wang J, Cao J, Wang S, Yao Z, Li W. IRDA: Incremental reinforcement learning for dynamic resource allocation. IEEE Trans Big Data 2022;8(3):770–83.

[27] He Y, Wang Y, Lin Q, Li J. Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks. IEEE Trans Veh Technol 2022;71(4):3495–506.

[28] Liang H, Zhang X, Hong X, Zhang Z, Li M, Hu G, Hou F. Reinforcement learning enabled dynamic resource allocation in the internet of vehicles. IEEE Trans Ind Inform 2021;17(7):4957–67.

[29] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.

[30] Co-Reyes JD, Miao Y, Peng D, Real E, Levine S, Le QV, Lee H, Faust A. Evolving reinforcement learning algorithms. 2022, ArXiv.

[31] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. Proc AAAI Conf Artif Intell 2016;30(1).

[32] Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd international conference on machine learning. Proceedings of machine learning research, Vol. 48, New York, New York, USA: PMLR; 2016, p. 1995–2003.

[33] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 2018.

[34] Yang J, Zhang J, Wang H. Urban traffic control in software defined internet of things via a multi-agent deep reinforcement learning approach. IEEE Trans Intell Transp Syst 2021;22(6):3742–54.

[35] Hu H, Jia X, He Q, Fu S, Liu K. Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. Comput Ind Eng 2020;149.

[36] Ying CS, Chow AHF, Chin KS. An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand. Transp Res B: Methodol 2020;140:210–35.

[37] Atallah RF, Assi CM, Khabbaz MJ. Scheduling the operation of a connected vehicular network using deep reinforcement learning. IEEE Trans Intell Transp Syst 2019;20(5):1669–82.

[38] Shin J, Sunwoo M. Vehicle speed prediction using a Markov chain with speed constraints. IEEE Trans Intell Transp Syst 2019;20(9):3201–11.

[39] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016, p. 1928–37.

[40] Zhang E, Masoud N. Increasing GPS localization accuracy with reinforcement learning. IEEE Trans Intell Transp Syst 2021;22(5):2615–26.

[41] Labao AB, Martija MAM, Naval PC. A3C-GS: Adaptive moment gradient sharing with locks for asynchronous actor-critic agents. IEEE Trans Neural Netw Learn Syst 2021;32(3):1162–76.

[42] Du J, Cheng W, Lu G, Cao H, Chu X, Zhang Z, Wang J. Resource pricing and allocation in MEC enabled blockchain systems: An A3C deep reinforcement learning approach. IEEE Trans Netw Sci Eng 2021.

[43] Jang H-C, Huang Y-C, Chiu H-A. A study on the effectiveness of A2C and A3C reinforcement learning in parking space search in Urban Areas Problem. In: 2020 international conference on information and communication technology convergence. ICTC, 2020, p. 567–71.

[44] Li H, Wu X, Liang Y, Zhang C. A multistakeholder approach to the airport gate assignment problem: Application of fuzzy theory for optimal performance indicator selection. Comput Intell Neurosci 2021;2021. Cited by: 0; All Open Access, Gold Open Access, Green Open Access.

[45] Huang S, Ontañón S. A closer look at invalid action masking in policy gradient algorithms. In: The international FLAIRS conference proceedings. Vol. 35, University of Florida George A Smathers Libraries; 2022.

[46] Kanervisto A, Scheller C, Hautamäki V. Action space shaping in deep reinforcement learning. In: 2020 IEEE conference on games (coG). 2020, p. 479–86.

[47] Müller A, Sabatelli M. Safe and psychologically pleasant traffic signal control with reinforcement learning using action masking. In: 2022 IEEE 25th international conference on intelligent transportation systems. ITSC, 2022, p. 951–8.

[48] Klar M, Glatt M, Aurich JC. Performance comparison of reinforcement learning and metaheuristics for factory layout planning. CIRP J Manuf Sci Technol 2023;45:10–25.

[49] CAAC. Technical guidelines for airport epidemic prevention and control. 8th ed.. 2021.

[50] p-christ. Deep-Reinforcement-Learning-Algorithms-with-PyTorch: PyTorch implementations of deep reinforcement learning algorithms and environments. 2024, https://github.com/p-christ/Deep-Reinforcement-Learning-Algorithms-with-PyTorch. [Accessed 03 October 2024].

[51] zhangzhengmin. GA for Gate assignment. 2024, https://github.com/zhangzhengmin/–GA. [Accessed 17 December 2024].

[52] Hu X-B, Di Paolo E. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: Multi-objective memetic algorithms. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009, p. 71–89.

[53] Ding C, Bi J, Wang Y. A hybrid genetic algorithm based on imitation learning for the airport gate assignment problem. Entropy 2023;25(4).