

Article

Semantic-Constrained Planning for Airport Vehicle Scheduling

Sheng Wang ^{1,2} and Tianhe Chi ^{1,2,*} ¹ Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China; wangsheng20@mailsucas.ac.cn² University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: chith@126.com

Abstract

As airport operations expand and ground handling becomes more complex, airport vehicle scheduling has evolved into a system-level decision problem constrained by operational rules, task dependencies, and resource availability. However, existing approaches largely rely on statistical correlation modeling and lack explicit representations of operational semantics and feasibility constraints, resulting in limited executability and poor cross-scenario robustness. To address this issue, we propose the Semantic-Constrained Planning Network (SCP-Net), which adopts a compile-first, plan-later paradigm by embedding operational semantics directly into the scheduling process. SCP-Net introduces an Operational Semantic Compiler (OSC) that encodes key flight task attributes, including service types, operational phases, and time windows, into a structured dependency representation, explicitly modeling task dependencies and task–vehicle feasibility relations. Based on this representation, a Constraint-Gated Planner (CGP) integrates operational dependencies and resource constraints through feasibility-aware gating, ensuring that planning is always conducted within valid operational regions. Through this design, SCP-Net directly generates schedules that are structurally consistent, semantically valid, and executable. Experimental results demonstrate that SCP-Net outperforms baseline methods in terms of executability, constraint violation rate, and cross-scenario stability, highlighting the effectiveness of explicit semantic modeling and constraint-driven planning for airport vehicle scheduling.

Keywords: airport vehicle scheduling; semantic-aware planning; constraint-guided decision making; structured dependency modeling; deep learning

1. Introduction

With the continuous expansion of airport operational scale, ground handling activities exhibit increasing complexity in task types, execution sequences, and resource coordination [1–4]. As a critical component of ground operations, airport vehicle scheduling must coordinate multiple task types and resources in dynamic environments to ensure the efficiency and stability of flight operations [5–7].

In recent years, scheduling methods based on deep learning and reinforcement learning have been widely applied to airport vehicle and ground service scheduling tasks. Related studies typically model the scheduling process as a state-to-decision mapping, generating scheduling solutions by learning patterns from historical data [8,9]. Under fixed scenarios or idealized conditions, such methods can improve efficiency and reduce costs and have therefore gradually become an important research direction in this field. However, from the perspective of real-world airport operations, these approaches still suffer from notable limitations [10,11].



Academic Editor: Keun Ho Ryu

Received: 23 January 2026

Revised: 11 February 2026

Accepted: 18 February 2026

Published: 6 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

On the one hand, most existing methods are centered on numerical state representations and mainly capture statistical correlations among tasks, vehicles, and time, while paying limited attention to the explicit modeling of task operational semantics [12,13]. For example, differences in the roles of various service tasks within the flight support process, precedence dependencies among tasks, and matching constraints between vehicle capabilities and service types are often encoded indirectly in high-dimensional features or reward signals [14]. This modeling paradigm requires the model to learn complex operational rules solely from data, which makes it sensitive to rule changes or data distribution shifts.

On the other hand, to enforce the feasibility of scheduling results, existing methods commonly rely on complex reward designs, penalty terms, or post hoc rule-based repair mechanisms [15–17]. These strategies increase model design complexity and make hyperparameter tuning more difficult, while also decouple scheduling constraints from the decision-making process itself, as constraints do not genuinely participate in decision computation [18]. In practical applications, this generate-then-repair workflow often fails to guarantee stable executability of scheduling solutions under complex constraints [19,20]. Therefore, it is necessary to develop a new modeling paradigm that allows scheduling models to explicitly perceive operational semantics and constraint structures during decision making, providing more direct structural support for stable and reasonable scheduling solutions [21].

To address the limitations of existing learning-based scheduling methods, where semantic information is often modeled implicitly and constraints rarely participate directly in decision computation, we propose the Operational Semantic Compiler (OSC) module. The OSC compiles raw flight and task data into structured and computable semantic representations. Unlike approaches that rely only on graph structures or attention mechanisms to model task correlations, the OSC uniformly encodes key operational attributes (e.g., service types, flight phases, and time windows) and explicitly builds task dependency relationships. This process forms an internal structured representation that reflects airport operational logic. As a result, the model can make decisions based on operational roles and priority relations, rather than approximating complex rules through statistical correlations or implicit feature aggregation.

Based on this semantic foundation, we further design a mechanism to continuously integrate constraints during schedule generation. Specifically, we propose the Constraint-Gated Planner (CGP) module. In CGP, operational dependencies and resource feasibility are directly embedded into decision score computation, instead of being treated as separate feasibility checks or post hoc repair steps. At each scheduling step, CGP dynamically evaluates whether candidate decisions are executable. It then restricts planning to a feasible solution space and performs continuous search within it, thereby filtering out infeasible solutions during decision generation. This constraint-aware planning paradigm is fundamentally different from methods that rely only on attention weight adjustment or posterior rule filtering, and it significantly enhances the executability and stability of the generated schedules.

The contributions of this paper are summarized as follows:

- (1) A semantics-driven scheduling modeling paradigm is proposed, which explicitly captures task semantics and operational dependency relationships in airport vehicle scheduling through structured representations.
- (2) A constraint-gated planning mechanism is designed, enabling operational constraints to participate endogenously in scheduling decision computation rather than relying on post hoc rule-based repairs.

- (3) An end-to-end deep learning-based scheduling model is constructed, providing a new modeling framework for addressing executability and stability challenges in airport vehicle scheduling under complex constraints.

2. Related Work

2.1. Airport Scheduling with Reinforcement Learning

In recent years, reinforcement learning has been widely applied to airport operational scheduling and ground control problems. Most studies model scheduling as a Markov decision process and learn policies through interaction with the environment. Varnousfaderani et al. [22] proposed a deep reinforcement learning-based vehicle scheduling algorithm for advanced air mobility, using dual dueling Q-networks to maximize air taxi revenue in both single-agent and multi-agent settings. This work shows the potential of reinforcement learning in complex scheduling tasks. However, its decision-making process depends heavily on reward function design, and operational rules are learned implicitly rather than explicitly represented during training. Ali et al. [23] later applied deep reinforcement learning to pushback rate control for mixed-mode runways. Their simulations indicate improvements in airside congestion and fuel consumption. However, this line of work focuses mainly on high-level traffic control, and has limited ability to model detailed task dependencies and resource matching relationships.

In energy- and resource-related scheduling, Al Mehadi and Watson [24] combined reinforcement learning with mixed-integer programming for electric vehicle charging in airport parking facilities. They analyzed trade-offs between efficiency and revenue under different strategies. However, this method mainly addresses energy allocation, where constraints are explicitly defined by an external optimization model. The reinforcement learning component itself still lacks explicit modeling of operational semantics. Caranti et al. [25] applied deep reinforcement learning to flight arrival sequencing, enabling dynamic priority management and improving key metrics under uncertainty. Nevertheless, their study focuses on flight sequencing rather than ground vehicle scheduling, and does not explicitly represent structural task dependencies. Similarly, Luo and Chen [26] used DQN for dynamic logistics scheduling and showed that reinforcement learning outperforms traditional methods in complex environments. However, scheduling is still treated as a state-action mapping, with relatively coarse representations of rules and dependency structures.

2.2. Learning-Based Task Allocation and Planning

Beyond reinforcement learning, several studies have applied machine learning and optimization methods to airport task allocation and path planning problems. Shadman et al. [27] proposed a bi-objective model that combines machine learning-based prediction with robust optimization for gate assignment under taxi-time uncertainty. By explicitly modeling uncertain parameters, their approach improves assignment stability. However, this method mainly addresses static or semi-static allocation problems and offers limited support for the continuous decision-making required in vehicle scheduling. Liu and Ding [28] integrated an improved Q-learning method with model predictive control for airport vehicle path replanning and obstacle avoidance. Their work emphasizes control precision and real-time response. However, it mainly focuses on vehicle motion control, and the scheduling logic still relies on externally defined rules.

In broader vehicle and automated guided vehicle (AGV) contexts, Yuan et al. [29] studied AGV path planning using multi-agent deep Q-networks and incorporated conflict resolution into the scheduling process. Nevertheless, their approach centers on path-level coordination, while task semantics and service phases are not systematically modeled. Ajayi [30] proposed a multi-layer optimization framework for sustainable airport oper-

ations, using machine learning to predict energy demand and support electric taxiing and charging planning. Although this study addresses system-level energy optimization, it does not directly consider task-level scheduling decisions. Dapaza [31] applied deep reinforcement learning to advanced air mobility communication and scheduling, jointly optimizing spectrum allocation and flight tasks through multi-agent models. While this work shows the scalability of learning-based approaches in complex systems, its scheduling objectives and constraint structures are substantially different from those in airport ground vehicle scheduling.

2.3. Constrained Planning Methods

In recent years, neuro-symbolic methods and diffusion models have brought new perspectives to constrained planning and sequential decision-making. These studies emphasize that explicit constraint modeling is crucial for improving feasibility and safety in complex scheduling tasks. Christopher et al. [32] integrated symbolic optimization into the generative process and showed that embedding constraints directly into decision computation improves safety in critical applications. This idea is particularly relevant to airport vehicle scheduling, where strict operational and safety rules must be satisfied. However, such methods usually depend on iterative sampling and optimization, which leads to high computational costs and makes them difficult to scale to large, discrete airport scheduling problems with complex temporal dependencies. Jung et al. [33] further argued that constraints should be incorporated directly into the planning process, rather than enforced through post hoc correction. However, their framework is mainly designed for continuous control tasks and has limited ability to represent discrete task dependencies and resource matching relationships.

In sequential decision-making research, Núñez Molina [34] improved constraint awareness using a neuro-symbolic planning architecture. Their results suggest that introducing explicit semantic rules can enhance decision stability, which is closely related to modeling service phases and task ordering in airport vehicle scheduling. Nevertheless, these approaches often rely on predefined symbolic rules, which reduces flexibility in dynamic operational environments.

In industrial scheduling, Fang et al. [35] applied generative models to handle integer constraints, showing the potential of generative approaches for complex scheduling problems. However, their method focuses on offline optimization and does not support dynamic task dependencies or real-time scheduling decisions.

3. Proposed Method

To overcome the limitations of existing airport vehicle scheduling methods in explicitly modeling task dependencies and resource feasibility under complex operational rules, which often lead to low executability and weak cross-scenario stability, we propose an end-to-end scheduling framework based on operational semantic constraints, called the Semantic-Constrained Planning Network (SCP-Net). The overall workflow of the proposed framework is shown in Figure 1.

Given flight information, task lists, and vehicle states within a rolling time window, the model first maps the raw inputs into a task set T and a vehicle set V , and extracts task semantics s_i , spatial attributes x_i , and vehicle capability descriptors u_v . The Operational Semantic Compiler then jointly encodes this information to generate compact semantic representations of tasks e_i , and further infers inter-task operational dependencies A_{ij} as well as the task–vehicle feasibility matrix F , thereby constructing a structured planning graph G . Based on G , the Constraint-Gated Planner takes the partially generated schedule P as a condition and incrementally computes scheduling preferences over candidate task–vehicle

pairs. Through feasibility gating, operational dependencies and resource constraints are dynamically injected into the decision-making process, yielding a constraint-aware decision distribution $p(i, v | P)$, from which the complete scheduling plan is generated sequentially.

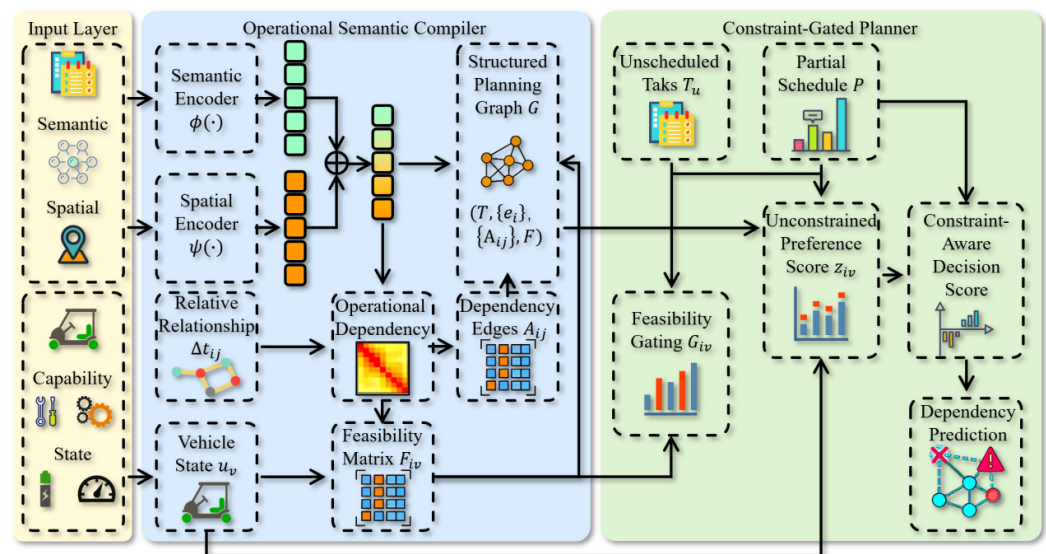


Figure 1. Overall architecture of the proposed Semantic-Constrained Planning Network.

3.1. Operational Semantic Compiler

The Operational Semantic Compiler aims to transform raw flight, task, and resource states into a computable, structured dependency representation, enabling the scheduling model to plan under explicit operational semantic constraints rather than relying solely on statistical correlations.

Let the task set within a rolling decision window be $T = \{1, \dots, N\}$ and the vehicle set be $V = \{1, \dots, M\}$. For each task $i \in T$, we unify its semantic attributes—such as service type, associated flight phase, time window, and priority—into a semantic vector s_i , and combine it with its spatial attributes x_i to construct a compact semantic representation e_i of task i , as shown in Equation (1):

$$e_i = \phi(s_i) + \psi(x_i) \in \mathbb{R}^d \quad (1)$$

Here, $\phi(\cdot)$ and $\psi(\cdot)$ are learnable encoding functions for semantic fields and spatial attributes, respectively, implemented as single-layer MLPs with non-shared parameters.

Based on the semantic representations of a task pair (i, j) , i.e., (e_i, e_j) , and their time-window difference Δt_{ij} , we learn the latent operational dependency r_{ij} between tasks, as defined in Equation (2):

$$l_{ij} = f_{\theta}([e_i; e_j; \Delta t_{ij}]), \quad r_{ij} = \text{softmax}(l_{ij}) \quad (2)$$

where $f_{\theta}(\cdot)$ is a fully connected scoring function. To explicitly inject operational dependencies into the structural representation, we gate the relation distribution using the time-window constraint Δt_{ij} , yielding a structured dependency representation between tasks, as shown in Equation (3):

$$A_{ij} = \sum_k r_{ij}[k] \cdot W_k \odot \Gamma(\Delta t_{ij}) \quad (3)$$

where W_k denotes the projection parameters for relation type k , $\Gamma(\cdot)$ is a feasibility gating function based on the time window, and A_{ij} constitutes the structural edge in the dependency graph.

Beyond inter-task dependencies, the compiler further characterizes the capability-matching relationship between tasks and vehicles to constrain the executable domain of subsequent scheduling decisions, as formulated in Equation (4):

$$F_{iv} = \sigma(g_\eta([e_i; u_v])) \quad (4)$$

where u_v represents the capability and state descriptor of vehicle v , and $F_{iv} \in (0, 1)$ indicates the soft feasibility of assigning task i to vehicle v .

Finally, the Operational Semantic Compiler outputs a structured planning graph G , which jointly encodes task semantics, inter-task dependencies, and task-vehicle feasibility, thereby providing explicit and computable operational semantic constraints for the subsequent planning module, as summarized in Equation (5):

$$G = (T, \{e_i\}, \{A_{ij}\}, F) \quad (5)$$

3.2. Constraint-Gated Planner

Based on the structured planning graph G , we further propose a Constraint-Gated Planner to generate executable scheduling solutions. The core idea is to inject operational constraints directly into the decision computation during planning, rather than performing post hoc repairs at the output stage.

Unlike approaches that generate schedules solely based on attention or scoring mechanisms, this module explicitly restricts the valid search space at each decision step by constructing a feasibility gate G_{iv} , thereby enabling continuous planning within a constrained feasible domain.

Let the set of unscheduled tasks in the current rolling window be $T_u \subseteq T$, and denote the partially generated schedule as P , which is initialized as an empty set. At each planning step, the model first computes an unconstrained preference score z_{iv} for each candidate task-vehicle pair (i, v) . This score reflects scheduling preference under the current context without directly encoding hard constraints, as shown in Equation (6):

$$z_{iv} = h_\omega(e_i, u_v, P) \quad (6)$$

where $h_\omega(\cdot)$ is a learnable fully connected planning network. Its inputs include the compact task semantic representation e_i , the vehicle state representation u_v , and the current partial plan P ; the output z_{iv} denotes the unconstrained preference score for assigning task i to vehicle v .

To explicitly incorporate operational constraints into the planning process, we construct a feasibility gate G_{iv} based on the dependency graph $\{A_{ij}\}$ and the task-vehicle feasibility matrix F , which characterizes the executability of a decision under the current state, as defined in Equation (7):

$$G_{iv} = F_{iv} \cdot \prod_{j \in T_u} \Gamma(A_{ji}, P) \quad (7)$$

where $\Gamma(\cdot)$ denotes a dependency-consistency gating function that checks whether the prerequisite dependencies of task i have been satisfied in the current partial schedule P . In airport vehicle scheduling, task-vehicle feasibility and task dependency satisfaction exhibit a “necessary condition” property, meaning that if any critical constraint is not satisfied, the corresponding decision is infeasible under the current state; therefore, a multiplicative gating formulation is adopted.

The gating term is then combined with the unconstrained preference score to obtain the final constraint-aware decision score $\tilde{z}_{iv}$, as shown in Equation (8):

$$\tilde{z}_{iv} = z_{iv} + \log(G_{iv} + \epsilon) \quad (8)$$

where ϵ is a numerical stability hyperparameter. Based on this score, the planner normalizes over candidates to select the next scheduling decision, gradually expanding the schedule, as given in Equation (9):

$$p(i, v | \mathcal{P}) = \text{softmax}(\tilde{z}_{iv}) \quad (9)$$

Here, $p(i, v | P)$ denotes the conditional probability of assigning task i to vehicle v as the next decision, given the current partial schedule P .

By introducing feasibility gating directly into the decision computation, constraints no longer exist merely as external rules or penalty signals, but instead participate explicitly in the formation of scheduling probabilities. This mechanism enables the planner to naturally generate schedules that satisfy operational constraints during both training and inference, thereby significantly improving executability and cross-scenario stability.

3.3. Loss Functions

We use two supervised objectives to complete training: Equation (10) constrains the sequential consistency of scheduling decisions, while Equation (11) improves the prediction accuracy of operational dependency structures.

$$\mathcal{L}_{\text{assign}} = -\sum_{t=1}^T \log p(i_t^*, v_t^* | P_{t-1}) \quad (10)$$

where t denotes the t -th step of rolling planning, P_{t-1} is the partially generated scheduling plan up to the previous step, $p(i, v | P_{t-1})$ is obtained by normalizing the constraint-aware scores $\tilde{z}_{iv}$ from the constraint-gated planner, and (i_t^*, v_t^*) is the corresponding ground-truth task–vehicle assignment in the supervised scheduling sequence.

$$\mathcal{L}_{\text{dep}} = -\sum_{i=1}^N \sum_{j=1}^N \sum_{k \in \mathcal{R}} y_{ij}[k] \log r_{ij}[k] \quad (11)$$

where $r_{ij} = \text{softmax}(l_{ij})$ is the task dependency distribution predicted by the operational semantic compiler, and y_{ij} denotes the supervised dependency labels. The final supervised objective is given in Equation (12):

$$\mathcal{L} = \mathcal{L}_{\text{assign}} + \lambda \mathcal{L}_{\text{dep}} \quad (12)$$

where λ is a balancing coefficient.

4. Experiments

4.1. Experimental Setup and Environment

We implemented the model and conducted a series of experiments on a unified platform to evaluate its performance. The operating system is Windows 11, equipped with an Intel Core i9-12900K processor, 64 GB DDR4 system memory, and an NVIDIA GeForce RTX 3090 GPU. SCP-Net is implemented in Python 3.9 using PyTorch 2.0.1 as the deep learning framework.

During training, the dimension d of task semantic representations in SCP-Net is fixed to 128. Both the semantic encoding network and the spatial encoding network in the Operational Semantic Compiler adopt a single-layer fully connected structure with ReLU activation. The decision scoring network in the Constraint-Gated Planner also follows a single-layer fully connected design. The batch size is set to 32, and the total number of

training epochs is 150. Model parameters are optimized using the Adam optimizer with a fixed learning rate of 1×10^{-3} , $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 1×10^{-5} . Dropout is applied after all fully connected layers with a ratio of 0.2. The weighting coefficient λ in the loss function is set to 0.5, and the numerical stability term ϵ is set to 1×10^{-8} .

4.2. Datasets and Evaluation Metrics

To systematically evaluate scheduling performance under different operational conditions, we adopt two airport ground service vehicle scheduling datasets with complementary characteristics. The first is a manually constructed synthetic benchmark dataset, AGSO-Bench [36], which includes four types of simulated airport configurations with different operational scales, exhibiting significant differences in daily flight volume, number of gates, and the scale of baggage and cargo handling facilities. Multiple independent instances are generated for each configuration. The data are organized at the flight level, recording flight arrival times, flight types, and parking durations. At the task level, required ground service operations, execution order constraints, and time-window limitations are provided. Vehicle constraints are modeled through capacity and service capability. Scheduling labels consist of feasible vehicle–task assignment solutions along with corresponding completion times and resource utilization information. The flight time distribution follows realistic 24 h airport operation patterns.

The second dataset, SZX-Real [37], is constructed based on a digital twin framework, integrating real airport operational data with simulation-generated data. By introducing disturbances such as flight delays, service duration fluctuations, and vehicle failures, it forms an event-driven temporal data structure to support continuous online rescheduling. The labels correspond to adaptive vehicle reassignment and service sequence adjustments under disturbed conditions. The data distribution changes over time and shows strong non-stationary characteristics.

In addition, we evaluate model performance from three complementary perspectives. Feasible Schedule Ratio (FSR) measures the proportion of generated schedules that satisfy all time, resource, and operational constraints, directly reflecting the model's ability to produce executable solutions. Average Constraint Violations (ACV) quantifies the average number of semantic, resource, or temporal constraint violations per schedule, providing a fine-grained assessment of the severity and frequency of infeasible decisions. Performance Variance (PV) across scenarios evaluates the stability of model performance under varying airport scales, flight densities, and random seeds, capturing the robustness and consistency of the scheduling policy across heterogeneous operational conditions.

4.3. Results and Analysis

4.3.1. Comparative Experiments

We compare the proposed SCP-Net with representative state-of-the-art methods on the AGSO-Bench and SZX-Real datasets, and the experimental results are reported in Table 1.

Overall, SCP-Net achieves the best performance on both datasets. DQN-based methods show low feasible schedule ratios and many constraint violations. These methods usually rely on limited state features and discrete action spaces, where constraints are only indirectly enforced through reward functions. When the number of constraints increases or the scenario changes, they struggle to adjust their decision structure in time, which makes them more likely to generate infeasible schedules. Although these models have relatively low computational complexity, their limited capacity restricts their ability to encode rich operational semantics, leading to poor executability in complex settings.

Table 1. Performance comparison between the proposed SCP-Net and representative methods on the AGSO-Bench and SZX-Real datasets.

Method	AGSO-Bench			SZX-Real			GFLOPs	Params (M)
	FSR (%) ↑	ACV ↓	PV ↓	FSR (%) ↑	ACV ↓	PV ↓		
DQN-Refuel [2]	78.42	1.87	0.093	71.36	2.43	0.128	1.8	3.5
Neural AGH [4]	83.65	1.42	0.071	75.82	2.01	0.104	4.2	9.2
PSO-QL-ARS [5]	81.73	1.58	0.079	73.94	2.18	0.112	3.0	6.1
GNN-DQN-JSSP [11]	84.21	1.36	0.066	77.10	1.92	0.097	7.8	14.8
MAMBA-Attn [16]	86.05	1.21	0.058	79.56	1.74	0.089	11.2	23.6
SCP-Net (Ours)	92.48	0.54	0.031	88.31	0.79	0.042	13.5	28.1

Neural AGH and GNN-DQN-JSSP partially improve task relationship modeling by introducing graph structures or attention mechanisms. However, their constraint representations are still implicitly embedded in features or loss functions, and they lack explicit feasibility control during decision making. These models require higher computational cost than basic DQN methods because of graph propagation and relational modeling. Yet, without constraint-aware decision gating, they achieve only moderate improvements in feasibility and stability, especially under dense dependencies or real-time disturbances.

MAMBA-Attn performs better than traditional reinforcement learning methods on both metrics, suggesting that sequence modeling and attention mechanisms help capture temporal correlations in scheduling. However, this improvement comes with increased computational complexity, reflected in higher GFLOPs and parameter counts. Since operational dependencies and resource constraints do not directly participate in decision computation, the model still produces a noticeable number of violating decisions in high-constraint scenarios.

In contrast, SCP-Net significantly increases the feasible schedule ratio and reduces the average number of constraint violations to the lowest level on both datasets. Although SCP-Net introduces additional computational cost, its overall complexity remains moderate and comparable to advanced sequence-based models. More importantly, the performance gains come from two principled design strategies rather than simple model scaling. First, the Operational Semantic Compiler explicitly transforms task semantics and operational dependencies into structured representations, so the model does not need to learn complex rules implicitly through excessive parameters. Second, the Constraint-Gated Planner integrates feasibility gating directly into the decision computation process, allowing constraints to influence decision probabilities and actively prevent infeasible solutions during generation. As a result, SCP-Net achieves a better balance between executability, stability, and computational efficiency, and maintains robust scheduling performance under complex constraints and dynamic disturbances.

4.3.2. Hyperparameter Analysis

To analyze the impact of the semantic representation dimension d on model performance, we conduct a hyperparameter study, and the results are shown in Figure 2.

It can be observed that when d increases from 64 to 128, the feasible schedule ratio on both datasets steadily improves, while the average number of constraint violations and performance variance across scenarios decrease. This suggests that a larger representation capacity helps the Operational Semantic Compiler encode task semantics and dependencies more reliably, and allows the Constraint-Gated Planner to make clearer feasibility distinctions during decision making, thereby reducing infeasible assignments and constraint conflicts. However, when d is further increased to 192, the improvement trend stops. Instead, a slight performance drop and higher variance appear on SZX-Real. This indicates

that overly large representation dimensions introduce redundant parameters, making the model more likely to learn unstable preference patterns under dynamic disturbances. As a result, the feasibility gating mechanism becomes less discriminative for candidate decisions, leading to smaller gains in feasible schedule ratio and greater fluctuations across scenarios.

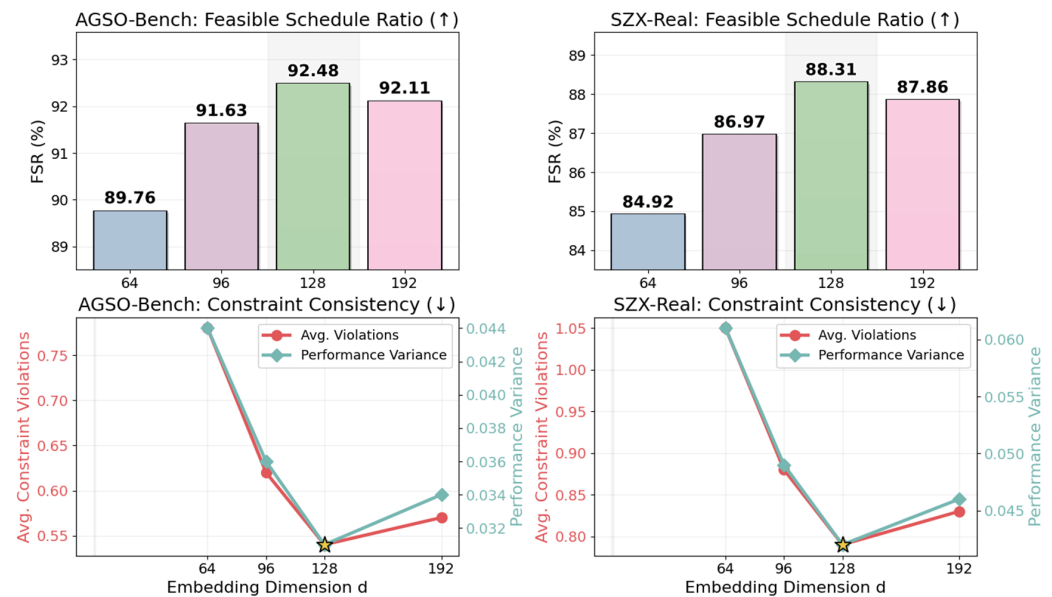


Figure 2. Sensitivity study of semantic embedding dimension d on AGSO-Bench and SZX-Real.

In addition, we further analyze the weighting coefficient λ in the loss function, and the experimental results are shown in Figure 3.

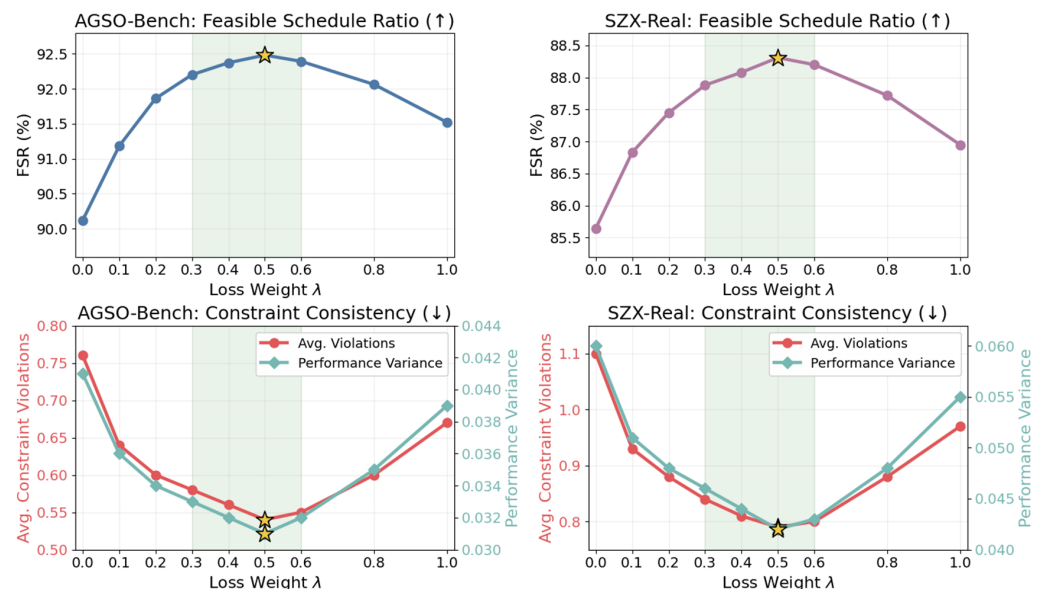


Figure 3. Sensitivity of λ in the loss function on AGSO-Bench and SZX-Real.

It can be observed that the overall performance is optimal when λ is set to 0.5. When λ is too small, training is almost dominated by L_{assign} , leading to insufficient learning of the dependency distribution r_{ij} . As a result, the structural consistency of the gating term G_{iv} is weakened, which manifests as a higher number of constraint violations and increased cross-scenario variability. Conversely, when λ is too large, the model overly emphasizes fitting dependency prediction, strengthening the influence of L_{dep} on parameter updates. This suppresses the planner's optimization of sequence assignment objectives, causing a

drift in which the predicted structure more closely matches the training labels while the allocation becomes less aligned with the current state. Ultimately, this leads to a decline in FSR and degraded stability.

4.3.3. Ablation Study

To verify the roles of the two core components in SCP-Net, we conduct ablation experiments by removing or replacing specific modules while keeping the remaining architecture unchanged, so as to evaluate the contribution of each component. The results are reported in Table 2.

Table 2. Ablation results of SCP-Net on AGSO-Bench and SZX-Real.

Variant	AGSO-Bench			SZX-Real		
	FSR (%) ↑	ACV ↓	PV ↓	FSR (%) ↑	ACV ↓	PV ↓
w/o OSC (no dependency graph, no F)	84.92	1.29	0.074	77.63	1.85	0.101
w/o CGP (no gate, preference-only planner)	86.47	1.11	0.068	79.54	1.62	0.092
w/OSC (remove time-window gate $\Gamma(\Delta t)$)	89.18	0.82	0.049	83.60	1.13	0.061
w/CGP (remove dependency-consistency gate $\Gamma(A_{ji}, P)$)	90.12	0.76	0.041	85.64	1.10	0.060
SCP-Net (Full)	92.48	0.54	0.031	88.31	0.79	0.042

From the ablation results, it can be observed that the complete SCP-Net achieves the best overall performance on both datasets. On AGSO-Bench, the full model attains a scheduling feasibility rate of 92.48, an average constraint violation count of 0.54, and a cross-scenario performance variance of 0.031; on SZX-Real, the corresponding values are 88.31, 0.79, and 0.042, respectively, indicating strong executability and stability.

In contrast, removing the Operational Semantic Compiler (OSC) leads to a clear performance degradation: on AGSO-Bench, the feasibility rate drops to 84.92, the violation count increases to 1.29, and the variance rises to 0.074; a similar trend is observed on SZX-Real, with a feasibility rate of 77.63, a violation count of 1.85, and a variance of 0.101. These results show that without explicitly modeling task semantics and operational dependencies, the planner is more likely to generate schedules that violate constraints.

When OSC is retained but the Constraint-Gated Planner (CGP) is removed, the performance is better than that of completely removing OSC, yet still significantly worse than the full model. On AGSO-Bench, the feasibility rate is 86.47, the violation count is 1.11, and the variance is 0.068; on SZX-Real, the corresponding values are 79.54, 1.62, and 0.092. This suggests that if constraints exist only implicitly and do not directly participate in decision score computation, the model struggles to maintain stable feasible decisions under complex dependency interactions or dynamic disturbances.

Further fine-grained ablation results indicate that removing either time-window gating or dependency-consistency gating leads to moderate performance drops. When time-window gating is removed, the violation count rises to 0.82 on AGSO-Bench and 1.13 on SZX-Real, and performance variance also increases on both datasets. Removing dependency-consistency gating performs better than completely removing CGP, but it still fails to match the full model in feasible schedule ratio and violation control. These results show that time-feasibility constraints and task-dependency constraints play com-

plementary roles, and removing either one reduces the model’s overall understanding of operational rules.

4.3.4. Visualization Experiments

To further understand the behavioral differences of various methods under complex constraint conditions, we analyze task dependency prediction heatmaps for DQN-Refuel, MAMBA-Attn, and SCP-Net under two representative operational scenarios, as shown in Figure 4.

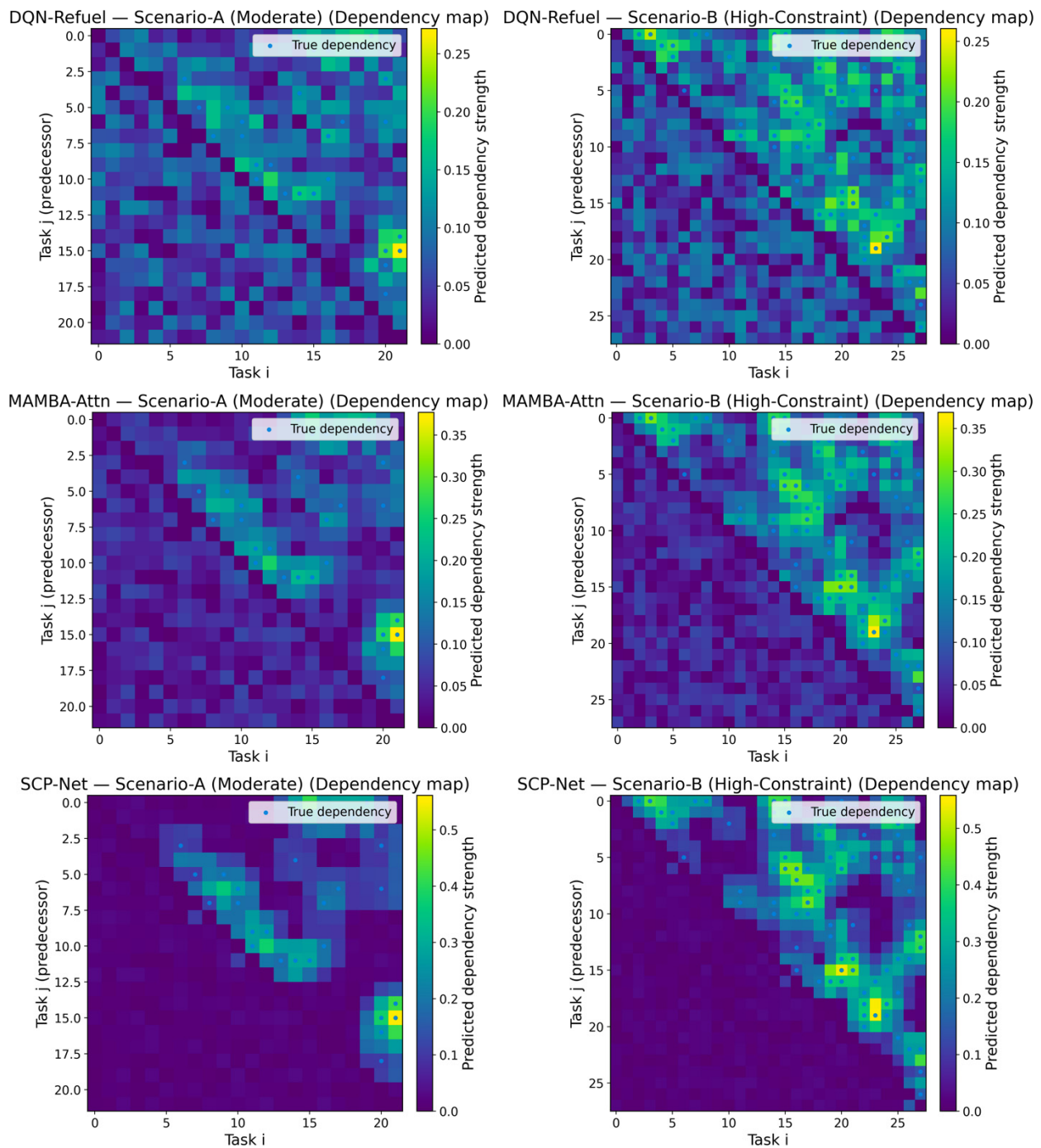


Figure 4. Visualization of predicted task dependency maps under moderate (Scenario-A) and high-constraint (Scenario-B) conditions. Each heatmap shows the predicted dependency strength between task pairs, while blue markers denote ground-truth dependencies.

Under Scenario-A (moderate constraints), DQN-Refuel exhibits a highly dispersed dependency prediction pattern. Although responses appear near some ground-truth dependencies, prediction strength spreads widely across the matrix, with non-dependent regions also receiving moderate responses. This indicates that the method primarily relies on reward signals to indirectly learn task relationships, making it difficult to form clear structural distinctions. Even under relatively low constraint pressure, its representation of task precedence remains ambiguous.

In the same scenario, MAMBA-Attn shows partial structural convergence, as high-response regions begin to focus on true dependencies. This suggests that sequence modeling and attention mechanisms can capture local temporal correlations. However, the predicted regions are still relatively broad, and dependency boundaries are not clearly defined. Some non-critical task pairs also show high responses, which indicates that constraint information is not explicitly used to filter predictions during modeling.

In contrast, SCP-Net produces a much more compact structural pattern in Scenario-A. High-intensity responses are almost entirely concentrated around ground-truth dependency points, while other regions remain close to zero. This demonstrates that, through structured representations generated by operational semantic compilation, the model can clearly distinguish task dependencies even under low-complexity conditions, thereby reducing interference from irrelevant dependencies.

Under Scenario-B (high constraints), the differences between methods become more evident. The heatmap of DQN-Refuel becomes almost uniform, with high-response areas spreading widely. True dependencies are hard to distinguish from the background, showing that the model struggles to maintain stable dependency representations when many constraints are stacked together. Although MAMBA-Attn can still detect some true dependencies, its overall structure deteriorates significantly. Many medium-strength responses appear in non-dependent regions, suggesting that its implicit modeling lacks effective feasibility control under high-constraint conditions.

SCP-Net, however, continues to maintain a clear dependency structure under high constraints. Prediction strength remains highly concentrated on true dependency locations, while infeasible regions are strongly suppressed and background regions exhibit consistently low responses. This suggests that operational dependencies are not merely learned passively through statistical correlations, but are continuously constrained during planning via semantic compilation and constraint-gated mechanisms, enabling SCP-Net to preserve structural consistency as dependency complexity increases.

5. Discussion

In SCP-Net, the Constraint-Gated Planner uses a multiplicative feasibility gating mechanism to enforce both operational dependencies and task–vehicle feasibility during decision making. This design effectively suppresses clearly infeasible assignments. However, it may also introduce some conservativeness in boundary cases. Specifically, when a task has a long dependency chain, the combined effect of multiple gating factors can significantly reduce its decision score, even if its prerequisite tasks are nearly completed and it will soon become executable. As a result, the task may be deprioritized too early, despite being practically feasible in the near future.

Similarly, when task–vehicle feasibility is represented by soft scores that are relatively low but still within an acceptable range (e.g., when a vehicle is about to become available or minor temporal adjustments could satisfy the constraints), multiplicative gating can amplify the combined suppressive effect of multiple soft constraints, thereby reducing exploration of these borderline-feasible candidates. As a result, although the mechanism improves overall schedule executability, it may also compress the search space around

boundary-feasible solutions under high uncertainty or tightly coupled temporal conditions, causing the planning behavior to exhibit a certain degree of conservativeness.

6. Conclusions

To address the challenge of ensuring both executability and cross-scenario stability in airport vehicle scheduling under complex operational rules and dynamic disturbances, we propose the Semantic-Constrained Planning Network (SCP-Net). SCP-Net follows a “compile-first, plan-later” strategy, which integrates operational semantics and feasibility constraints directly into the scheduling process. Specifically, SCP-Net uses an Operational Semantic Compiler (OSC) to explicitly transform task semantics, task dependencies, and task–vehicle feasibility into structured representations. It then introduces a Constraint-Gated Planner (CGP) to apply feasibility gating during decision making. This design allows the model to generate scheduling decisions within a constrained feasible space, thereby reducing infeasible assignments and constraint conflicts.

Across two complementary benchmarks, AGSO-Bench and SZX-Real, SCP-Net achieves the best overall performance. Comparative results show that SCP-Net attains scheduling feasibility rates of 92.48% and 88.31% on the two datasets, respectively, while reducing the average constraint violation counts to 0.54 and 0.79, and achieving the lowest performance variance in cross-scenario stability metrics. Compared with reinforcement learning methods that rely on reward-based indirect constraints, and sequence/attention-based methods that primarily depend on implicit feature fusion, SCP-Net maintains more stable feasible decisions under high-constraint and non-stationary disturbance conditions. This demonstrates that explicit operational semantic modeling and constraint-gated planning can more directly suppress the generation of infeasible allocations.

Further hyperparameter and ablation experiments confirm the importance of the key design choices. A moderate semantic representation dimension improves dependency encoding and strengthens feasibility discrimination during planning. In contrast, removing either OSC or CGP results in lower feasible schedule ratios, more constraint violations, and reduced stability. In addition, dependency-consistency gating and time-window gating show complementary benefits in high-constraint scenarios, and removing either one weakens the model’s overall understanding of operational rules. Visualization results further demonstrate that SCP-Net produces dependency predictions that are more concentrated around ground-truth relations with lower background noise, especially under high-constraint conditions. This reflects the continuous constraining effect of its structured semantic compilation and feasibility gating mechanisms.

Despite the significant improvements in executability and stability, SCP-Net has a limitation inherent to its mechanism. The multiplicative feasibility gating in CGP may introduce overly strong suppression when dependency chains are long or soft feasibility scores are low, causing some borderline-feasible candidate decisions to be prematurely down-weighted. This can reduce the exploration space and introduce a degree of conservativeness. Future work will explore smoother gating calibration strategies and incorporate dependency confidence estimation and hierarchical constraint handling, enabling the model to better adapt to complex disturbances and boundary-feasible solutions while maintaining low violation rates.

Author Contributions: S.W.: Writing—Original Draft, Writing—Review & Editing, Data Curation, Formal analysis; T.C.: Writing—Original Draft, Writing—Review & Editing, Methodology, Software. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: During the manuscript writing stage, an artificial intelligence-assisted tool ChatGPT 5.2. was used for language polishing to improve clarity and readability. The tool was not involved in the research design, experiments, or result analysis, and all scholarly content of this paper was completed by the authors, who take full responsibility for it.

Conflicts of Interest: The authors declare that they do not have any known interests or personal relationships that could potentially influence the reported work in this paper.

References

- Vieira, M.; Vieira, M.A.; Galvão, G.; Louro, P.; Fantoni, A.; Vieira, P.; Véstias, M. Enhancing Airport Traffic Flow: Intelligent System Based on VLC, Rerouting Techniques, and Adaptive Reward Learning. *Sensors* **2025**, *25*, 2842. [\[CrossRef\]](#) [\[PubMed\]](#)
- Chen, W.; Li, Y. Study on dynamic scheduling method of airport refueling vehicles based on DQN. *Xibei Gongye Daxue Xuebao J. Northwest. Polytech. Univ.* **2024**, *42*, 764–773. [\[CrossRef\]](#)
- Wang, X. Intelligent Vehicle Scheduling for Green Airport Baggage Transport Service. Master's Thesis, The Hong Kong Polytechnic University, Hong Kong, 2025.
- Wu, Y.; Zhou, J.; Xia, Y.; Zhang, X.; Cao, Z.; Zhang, J. Neural airport ground handling. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 15652–15666. [\[CrossRef\]](#)
- Zhu, J.; Han, G.; Shang, P.; Zhao, Z.; Huang, H.; Xiao, F. A Q-Learning based Particle Swarm Optimization for Aircraft Routing and Scheduling in Airport Terminal Area. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 22609–22624. [\[CrossRef\]](#)
- Zhou, J.; Wu, Y.; Cao, Z.; Song, W.; Zhang, J.; Chen, Z. Learning large neighborhood search for vehicle routing in airport ground handling. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 9769–9782. [\[CrossRef\]](#)
- Yuan, Y.; Han, Y.; Xue, J.; Guo, Y. A Dynamic Dispatching Method in the Unmanned Airport Baggage Transportation System. In Proceedings of the 2022 23rd IEEE International Conference on Mobile Data Management (MDM), Paphos, Cyprus, 6–9 June 2022; IEEE: New York, NY, USA, 2022; pp. 500–505.
- Xiang, Z.; Sun, H.; Zhang, J. Application of improved Q-learning algorithm in dynamic path planning for aircraft at airports. *IEEE Access* **2023**, *11*, 107892–107905. [\[CrossRef\]](#)
- Cai, K.; Li, Z.; Guo, T.; Du, W. Multiairport departure scheduling via multiagent reinforcement learning. *IEEE Intell. Transp. Syst. Mag.* **2023**, *16*, 102–116. [\[CrossRef\]](#)
- Ali, H.; Pham, D.T.; Alam, S.; Schultz, M. A deep reinforcement learning approach for airport departure metering under spatial–temporal airside interactions. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 23933–23950. [\[CrossRef\]](#)
- Liu, Z.; Wang, Y.; Liang, X.; Ma, Y.; Feng, Y.; Cheng, G.; Liu, Z. A graph neural networks-based deep Q-learning approach for job shop scheduling problems in traffic management. *Inf. Sci.* **2022**, *607*, 1211–1223. [\[CrossRef\]](#)
- Nguyen-Duy, A.; Pham, D.T.; Lye, J.Y.; Ta, D. Reinforcement Learning for Strategic Airport Slot Scheduling: Analysis of State Observations and Reward Designs. In Proceedings of the 2024 IEEE Conference on Artificial Intelligence (CAI), Singapore, 25–27 June 2024; IEEE: New York, NY, USA; 2024; pp. 1195–1201.
- Kumar, P.K.; Witter, J.; Paul, S.; Dantu, K.; Chowdhury, S. Graph learning based decision support for multi-aircraft take-off and landing at urban air mobility vertiports. In Proceedings of the AIAA Scitech 2023 Forum, National Harbor, MD, USA, 23–27 January 2023; p. 1848.
- Zheng, Y.; Guo, R.; Liu, G. Combining Graph Neural Networks and Deep Reinforcement Learning for Aircraft Maintenance Stand Scheduling. In Proceedings of the 2024 8th International Conference on Computer Science and Artificial Intelligence, Beijing, China, 6–8 December 2024; pp. 333–339.
- Paul, S.; Witter, J.; Chowdhury, S. Graph Learning-Based Fleet Scheduling for Urban Air Mobility Under Operational Constraints, Varying Demand & Uncertainties. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, Ávila, Spain, 8–12 April 2024; pp. 638–645.
- Li, J.; Fu, W.; Huang, G.; Liu, K.; Zhang, J.; Fu, Y. Combining MAMBA and Attention-Based Neural Network for Electric Ground-Handling Vehicles Scheduling. *Systems* **2025**, *13*, 155. [\[CrossRef\]](#)
- Yang, W.; Liu, H.; Jia, A.; Chen, H.; Yang, H. A reinforcement learning-based routing approach for optimizing airport ground movement. *Transp. B Transp. Dyn.* **2025**, *13*, 2496824. [\[CrossRef\]](#)
- Singh, N.; Akcay, A.; Dang, Q.V.; Martagan, T.; Adan, I. Dispatching AGVs with battery constraints using deep reinforcement learning. *Comput. Ind. Eng.* **2024**, *187*, 109678. [\[CrossRef\]](#)
- Shen, J.; Guo, Q.; Lin, L.; Zhong, Y.; Deng, L. Taxiway path planning at airports based on dynamic directed graphs and DQN. *Aerosp. Sci. Technol.* **2025**, *168*, 111295. [\[CrossRef\]](#)

20. Shahbazian, R.; Pugliese, L.D.P.; Guerriero, F.; Macrina, G. Integrating machine learning into vehicle routing problem: Methods and applications. *IEEE Access* **2024**, *12*, 93087–93115. [[CrossRef](#)]
21. Yuan, D.; Zhong, Y.; Zhu, X.; Chen, Y.; Jin, Y.; Du, X.; Tang, K.; Huang, T. Trajectory Planning for Unmanned Vehicles on Airport Apron Under Aircraft–Vehicle–Airfield Collaboration. *Sensors* **2024**, *25*, 71. [[CrossRef](#)] [[PubMed](#)]
22. Varnousfaderani, E.S.; Shihab, S.A.; Dulia, E.F. DeepDispatch: Deep Reinforcement Learning-Based Vehicle Dispatch Algorithm for Advanced Air Mobility. *J. Air Transp.* **2025**, *33*, 26–47. [[CrossRef](#)]
23. Ali, H.; Pham, D.T.; Alam, S. Toward greener and sustainable airside operations: A deep reinforcement learning approach to pushback rate control for mixed-mode runways. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 18354–18367. [[CrossRef](#)]
24. Al Mehadi, A.; Watson, O.; Kamineneni, A. Comparative Analysis of MILP and RL Algorithms for EV Charging Scheduling at Airport Parking. In Proceedings of the 2024 IEEE Kansas Power and Energy Conference (KPEC), Manhattan, KS, USA, 25–26 April 2024; IEEE: New York, NY, USA; 2024, pp. 1–6.
25. Caranti, L.; Ribeiro, M.; Carré, M. Dynamic Arrival Prioritization With Target Time Management and Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 20748–20765. [[CrossRef](#)]
26. Luo, X.; Chen, Z. Research on Dynamic Scheduling Optimization for Dongguan Smart Logistics Resources Using Reinforcement Learning. In Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence, Dongguan, China, 28–30 March 2025; pp. 964–968.
27. Shadman, M.; Kordani, A.A.; Akbaripour, H. A machine learning approach to address robust bi-objective gate assignment under aircraft’s taxi delay uncertainties. *Expert Syst. Appl.* **2025**, *285*, 128055. [[CrossRef](#)]
28. Liu, X.; Ding, Y. Airport Vehicle Scheduling and Obstacle Avoidance Predictive Control Based on Improved Q-Learning and Trajectory Re-Planning. In Proceedings of the 3rd International Conference on Internet of Things and Smart City (IoTSC 2023), Chongqing, China, 24–26 March 2023; SPIE: Bellingham, WA, USA, 2023; Volume 12708, pp. 765–775.
29. Yuan, M.; Lu, S.; Zheng, L.; Yu, Q.; Pei, F.; Gu, W. Research on AGV path planning in flexible job-shop scheduling problem based on multi-agent deep Q network. *Intell. Serv. Robot.* **2025**, *18*, 709–722. [[CrossRef](#)]
30. Ajayi, A. Future Grid for a Sustainable Green Airport: Meeting the New Loads of Electric Taxiing and Electric Aircraft. Ph.D. Thesis, Cranfield University, Cranfield, UK, 2023.
31. Dapaza, R.D. Spectrum Optimization for Advanced Air Mobility Communications Using Deep Reinforcement Learning. Ph.D. Thesis, University of Louisville, Louisville, KY, USA, 2024.
32. Christopher, J.K.; Cardei, M.; Liang, J.; Fioretto, F. Neuro-Symbolic Generative Diffusion Models for Physically Grounded, Robust, and Safe Generation. *arXiv* **2025**, arXiv:2506.01121. [[CrossRef](#)]
33. Jung, W.; Mishra, U.A.; Arachchige, N.R.; Chen, Y.; Xu, D.; Kousik, S. Joint Model-based Model-free Diffusion for Planning with Constraints. *arXiv* **2025**, arXiv:2509.08775.
34. Núñez Molina, C. Application of Neuro-Symbolic Artificial Intelligence to Sequential Decision Making. Doctoral Dissertation, Universidad de Granada, Granada, Spain, 2025.
35. Fang, W.; Du, W.; He, R.; Tang, Y.; Jin, Y.; Yen, G.G. Diffusion model-based multiobjective optimization for gasoline blending scheduling. *IEEE Comput. Intell. Mag.* **2024**, *19*, 61–76. [[CrossRef](#)]
36. Cai, Z.; Gao, W.; Feng, R.; Li, Y.; Xu, M. Benchmark for the scheduling problems of airport ground support operations and a case study. *Appl. Soft Comput.* **2025**, *169*, 112555. [[CrossRef](#)]
37. Luo, Q.; Liu, H.; Liu, C.; Deng, Q. Multi-strategy cooperative scheduling for airport specialized vehicles based on digital twins. *Sci. Rep.* **2024**, *14*, 15533. [[CrossRef](#)]

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