Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Neural construction heuristics for large-scale airport ground scheduling

Yang Zhang
The Second Research Institute of Civil Aviation Administration of China
Mingjie Tang
The Second Research Institute of Civil Aviation Administration of China
Hua Cheng
The Second Research Institute of Civil Aviation Administration of China
Jizhen Ma
The Second Research Institute of Civil Aviation Administration of China
Runhao Xu
The Second Research Institute of Civil Aviation Administration of China
Fei Xiang
xiangfei@caacsri.com
The Second Research Institute of Civil Aviation Administration of China

Article

Keywords: Airport ground scheduling, Multi-depot vehicle routing problem with time windows, Large-scale routing problem, Neural construction heuristic algorithm, Reinforcement learning

Posted Date: September 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10761752/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Neural construction heuristics for large-scale airport ground scheduling

Yang Zhang[1,+], Mingjie Tang[1,+], Hua Cheng[1], Jizhen Ma[1], Runhao Xu[1], and Fei Xiang[1,*]

[1]The Second Research Institute of Civil Aviation Administration of China, Civil Aviation Chengdu Information Technology Co., Ltd, Chengdu, 610041, China
[*]xiangfei@caacsri.com
[+]these authors contributed equally to this work

ABSTRACT

Airport ground scheduling (AGS) is crucial for flight operation support and can significantly improve the overall operational efficiency of airports. Generally, AGS can be universally formulated as the Multi-depot capacitated vehicle routing problem with time windows (MDCVRPTW). For hub airports, the corresponding MDCVRPTW features an extremely large scale and high complexity, which severely limits the solution effectiveness of traditional approaches. In this paper, we propose an efficient neural construction heuristic algorithm, which is capable of effectively solving large-scale MDCVRPTW instances. Specifically, we decompose the solution process into two phases. In the first phase, the MDCVRPTW is transformed into a Single-Depot capacitated vehicle routing problem with time windows (SDCVRPTW) by introducing a virtual depot, and then solved using the Transformer-based model trained by reinforcement learning (RL). In the second phase, a customized heuristic algorithm is designed to allocate the obtained solutions according to the distance between vehicle routes and real depots, assigning all routes to their corresponding depots and deriving the final solution to the original MDCVRPTW. Extensive experiments conducted on the real-world scenario of a hub airport in Southwest China demonstrate that the proposed method outperforms existing open-source solvers, meta-heuristic algorithms, and state-of-the-art approaches, which verifies its effectiveness in solving large-scale MDCVRPTW. In addition, the test results on real operational data also indicate that our method has great potential for practical deployment and application.

Keywords: Airport ground scheduling, Multi-depot vehicle routing problem with time windows, Large-scale routing problem, Neural construction heuristic algorithm, Reinforcement learning

Introduction

Airport ground scheduling (AGS) constitutes a critical component of flight operation support which can effectively reduce flight delays and related economic losses, and improve the overall operational efficiency of the airports. AGS encompasses dozens of distinct flight service tasks, such as shuttle bus, luggage, refueling and catering. These tasks share universal characteristics (e.g., time windows, demand), and thus can be universally formulated as a Multi-depot capacitated vehicle routing problem with time windows (MDCVRPTW). For hub airports, the daily flight volume ranges approximately from 800 to 1,200 departures and arrivals, rendering the scale of the corresponding MDCVRPTW extremely large.

Given the combinatorial nature of the problem, exact algorithms have been widely adopted for its solution. These algorithms typically formulate AGS as a Mixed-Integer Linear Programming (MILP) model[1,2] and then leverage mature solvers (e.g., commercial solvers[3]) for solution. However, the computational complexity of solving the entire problem are particularly high, such solvers are unable to obtain optimal solutions within a reasonable time frame. Therefore, as alternatives to balance solution quality and solution time, traditional metaheuristic algorithms or improved heuristic algorithms are also used to solve AGS[4–7]. Nevertheless, they heavily rely on domain expertise to design high-cost hand-crafted rules. More critically, when confronted with large-scale AGS tasks, their solution time still fails to meet practical operational requirements, which restricts their applicability in real-world airport scenarios.

In recent years, a substantial number of deep learning methods have been proposed to address the vehicle routing problem (VRP) and its relevant variants. Most of these neural learning-based methods follow the Transformer-based architecture and are trained based on reinforcement learning (RL)[8–10]. They are capable of generating high-quality solutions in a short time frame through sequential decision-making processes, and can well adapt to large-scale combinatorial optimization problems[11–13]. However, due to their inherent encoder-decoder architectural characteristics, these methods cannot directly solve the MDCVRPTW , but are only applicable to the single-depot vehicle capacitated routing problem with time windows (SDCVRPTW), which means the sequential decision-making mechanism of such models only accounts for the service routes of

vehicles departing from a single depot, and cannot accommodate the simultaneous dispatch of vehicles from multiple depots. As a result, they still exhibit inherent limitations when applied to practical AGS tasks.

In this paper, we propose a neural construction heuristic algorithm to address the large-scale MDCVRPTW in AGS. First, we uniformly formulate the AGS task as a MDCVRPTW and standardize it into a MILP model, so as to facilitate systematic problem analysis and solution derivation. Further, we design a two-stage neural construction heuristic algorithm. In the first stage, the MDCVRPTW is transformed into a SDCVRPTW, which is efficiently solved by an improved Transformer-based model trained with RL. In the second stage, a heuristic algorithm is developed to allocate the obtained solutions to corresponding depots, deriving the final solution to the MDCVRPTW. We demonstrate that the proposed method can effectively tackle large-scale MDCVRPTW instances and generate high-quality solutions, outperforming existing state-of-the-art approaches. Our primary contributions are summarized as follows:

- We conceptualize the AGS task as an MDCVRPTW and establish a unified mathematical model for it, enabling sequential solution generation using the proposed neural construction heuristic algorithm.
- We break down the solving process into two steps. In the first step, we parameterize the proposed neural learning-based method with an Transformer-based policy network which is trained with RL to sequentially generate intermediate solutions. In the second step, we develop a heuristic algorithm to efficiently obtain the final solutions of MDCVRPTW. To the best of our knowledge, this is the first work to solve large-scale MDCVRPTW in AGS with neural methods.
- Extensive experiments were conducted on the real-world scenarios of a hub airport in Southwest China which demonstrate that our proposed method can efficiently generate high-quality solutions for large-scale MDCVRPTW instances, and its performance outperforms a series of meta-heuristic algorithms and state-of-the-art neural learning-based approaches.

Related Work

Airport Ground Scheduling

AGS refers to a series of support task operations conducted in airport operation control for arriving flights. Typically, these support tasks are heterogeneous and possess distinct task attributes. Most existing studies on AGS focus on a specific type of support task, modeling it as VRP or its relevant variants based on the task's unique characteristics. Among these efforts, Du et al.[14] formulated the scheduling of aircraft towing operations as the Vehicle Routing Problem with Time Windows (VRPTW), and solved it using a column generation approach based on the MILP formulation. Han et al.[15] described the scheduling of passenger shuttle vehicles in the form of an MILP model, which was solved via CPLEX with the objective of minimizing the number of deployed vehicles. Norin et al.[16,17] also modeled de-icing operation scheduling as the VRPTW, and adopted the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm to minimize both the total travel distance of de-icing vehicles and the overall flight delay time. Guo et al.[18] addressed the vehicle scheduling problem for airport baggage transportation, and proposed a solution leveraging parameter or rule selection within the genetic algorithm framework. Meanwhile, a growing body of research has been dedicated to refueling operation scheduling, where meta-heuristic algorithms such as Ant Colony Optimization (ACO) and Iterated Local Search (ILS) are commonly employed for problem resolution[19–21].

On the other hand, there exist a limited number of studies focusing on the global optimization of multiple types of tasks in AGS scenarios. Padrón et al.[22] proposed a decomposition method based on Constraint Programming (CP) and Large Neighborhood Search (LNS) to schedule multiple types of ground support vehicle tasks. Building upon this work, they[23] further improved the method by solving the decomposed VRPTW instances in different sequences, whose performance is affected by the interaction relationships between various ground operations. Liu et al.[24] adopted the Non-dominated Sorting Genetic Algorithm II (NSGA-II) to tackle the routing problem involving multiple types of vehicle fleets, where a separate VRPTW model was established for each fleet. The dual objectives of this method are to minimize the total number of vehicles deployed and the time cost associated with the assigned vehicles. Additionally, Zhou et al.[25] introduced an Adaptive Large Neighborhood Search (ALNS) algorithm integrated with Graph Convolutional Network to tackle the multi-operation scheduling challenge in AGS. Follow this work, Wu et al.[26] designed a task-specific reinforcement learning algorithm targeting multi-operation flight ground support tasks in AGS. Their method employs a combined execution paradigm, where operations with varying priorities are processed in a serial manner and those with equivalent priorities are executed in parallel, ensuring the stable and efficient operation of the entire AGS process.

Neural Learning-based Methods for VRP

With the rapid development of deep neural networks, a large number of neural learning-based methods for VRP have emerged in recent years. Kool et al.[27] proposed a Transformer-based architecture named Attention Model (AM) with RL algorithm to solve Neural Combinatorial Optimization (NCO) including VRP. The AM can obtain promising performance on various small-scale NCO. Therefore, the AM has become the cornerstone of neural learning-based methods, and many variations of AM models

have been proposed to solve for NCO. Kwon et al.[28] proposed an improved Policy Optimization with Multiple Optima (POMO) based on AM, which aims to utilize the symmetry in the representation of combinatorial optimization solutions and adopts an improved REINFORCE algorithm to promote diversified sampling towards all optimal solution directions. In addition, a new enhanced inference method is introduced, which can be well used in conjunction with POMO. Kim et al.[29] article proposed a novel hierarchical problem solving strategy called learning collaborative policies (LCP), which can effectively find approximate optimal solutions for VRP through two iterative deep reinforcement learning strategies (seeder and modifier). Xin et al.[30] proposed a Multi-Decoder Attention Model (MDAM) to train multiple different strategies, which effectively improves the probability of finding high-quality solutions compared to existing methods that only train a single strategy. In addition, a customized beam search strategy was designed to fully utilize the diversity of MDAM.

However, these neural construction methods still exhibit extremely poor generalization performance on large-scale problem instances. Directly training such models for large-scale problems is highly challenging, if not entirely infeasible. To address these issues, a number of neural learning-based approaches have recently been proposed to handle large-scale VRP instances[11–13,31,32]. Despite their effectiveness in solving large-scale VRP instances, these approaches fail to directly tackle the MDCVRPTW due to an intrinsic limitation of their encoder-decoder architecture, which renders them notably inadequate for solving the MDCVRPTW.

Multi-Depot Capacitated Vehicle Routing Problem with Time Windows

The MDCVRPTW is a classic variant of the VRP and has been widely applied to practical scenarios in the transportation domain. As a NP-hard problem, the MDCVRPTW cannot be solved with exact algorithms in a computationally feasible manner even for relatively small-scale instances[33]. For this reason, existing research has primarily focused on developing heuristic and meta-heuristic algorithms to address this problem. A common solution strategy for the MDCVRPTW is to decompose it into multiple SDCVRPTW and adopt a two-stage “cluster-first, route-second" framework for resolution[34]. Gu et al.[35] simplified the MDCVRPTW into a single-depot VRPTW via a depot clustering method and solved the transformed problem using a bee colony algorithm. Tohidifard et al.[36] proposed a hybrid approach combining genetic algorithms and particle swarm optimization to tackle the MDCVRPTW. Xue[37] designed a two-stage framework integrating the k-means algorithm and an adaptive ant colony optimization algorithm for MDCVRPTW solution. Dursun et al.[38] developed a novel hybrid genetic algorithm with variable neighborhood search (HGA-VNS) to solve the MDCVRPTW with a heterogeneous fleet of vehicles. They analyzed the problem by defining narrow and wide time windows, which correspond to the fulfillment of urgent and regular service demands respectively, and incorporated real-world aviation and highway operation data into their model. Notably, several critical aviation-specific constraints were also embedded in the model, including aircraft range, landing and take-off (LTO) costs determined by engine types, and penalty costs for constraint violations. Nevertheless, the computational efficiency and performance of the aforementioned methods are severely challenged when confronted with large-scale MDCVRPTW instances. To address the limitations of existing studies, this work designs a two-stage neural construction heuristic algorithm based on the Transformer architecture, which is tailored to solve the large-scale MDCVRPTW in AGS. As we know, this is among the earliest efforts that have achieved effective resolution of large-scale AGS tasks with deep learning method.

MILP Model Formulation

Firstly, we formulate the MDCVRPTW in AGS as a MILP model with flight demand. Consider an undirected graph $G = (V, E)$ where V represents a node set and $E = \{(i, j)|i, j \in V; i \neq j\}$ represents an edge set respectively. V consists of two disjoint subsets: $V_f = \{1, 2, \ldots, n\}$ represents n flight nodes with demand d_i, $i \in V_f$; $V_d = \{n+1, n+2, \ldots, n+m\}$ represents m depot nodes. Each edge $(i, j) \in E$ is associated with a travel cost c_{ij} and the cost represents distance for this paper. The problem involves a fleet of K homogeneous vehicles with identical capacity W. For each vehicle k, the following parameters apply: p_i denotes the service time required for flight i, t_{ij} is the transfer time from flight i to j and $[a_i, b_i]$ represents the time window during which service for flight i must begin. The problem includes two categories of decision variables, 1) x_{ijk} is a binary variable, which equals to 1 if vehicle k traverses (i, j), and 0 otherwise; 2) Continuous temporal variables T_{ik} representing the service start time at flight i by vehicle k. The objective is to minimize the total travel cost:

$$\min. \quad \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in V}\sum_{k\in K} x_{ijk} = 1, \forall j \in V_f \tag{2}$$

$$\sum_{i\in V_f} d_i \sum_{j\in V} x_{ijk} \le W, \forall k \in K \tag{3}$$

$$\sum_{i\in V} x_{iuk} = \sum_{j\in V} x_{ujk}, \forall u \in V_f, k \in K \tag{4}$$

$$\sum_{j\in V_f} x_{djk} = \sum_{i\in V_f} x_{idk}, \forall k \in K, d \in V_d \tag{5}$$

$$T_{jk} \ge T_{ik} + p_i + t_{ij} - M \cdot (1 - x_{ijk}), \forall (i,j) \in E, k \in K \tag{6}$$

$$a_i \le T_{ik} \le b_i, \forall k \in K, i \in V_f \tag{7}$$

The objective function expressed in Eq.(1) is designed to minimize the total travel distance. Eq.(2) guarantees that each flight is assigned to exactly one vehicle. Eq.(3) enforces that the total demand serviced by a vehicle does not exceed its maximum capacity. Eq.(4) ensures route continuity by requiring that a vehicle arriving at a flight must also depart from it. Eq.(5) specifies that all vehicles must start and end their routes at the same depot. Eq.(6) and (7) ensure that service begins within the designated time windows and that temporal continuity is maintained throughout the route, where M is a large constant value(i.e. 10^4).

Methods

We model the AGS problem as a MDCVRPTW and solve it via a two-phase strategy. In the first phase, the MDCVRPTW is transformed into a SDCVRPTW by introducing a virtual depot, and an improved Transformer-based model is applied to obtain vehicle scheduling results departing from the virtual depot. In the second phase, a heuristic algorithm is designed to assign all vehicle routes to corresponding real depots based on the proximity between the start/end points of each route and the depots, yielding the final solution for the MDCVRPTW.

For the transformation process of the MDCVRPTW into a SDCVRPTW, a virtual depot is first established. All flight service points are regarded as service nodes subordinate to this virtual depot, while real depots are temporarily excluded from the initial modeling. Subsequently, all vehicles are scheduled to depart from this virtual depot and travel to respective service points to complete AGS tasks. Notably, the costs (including time consumption and travel distance) incurred during the travel from the virtual depot to each service point are set to zero. Through the aforementioned transformation, the MDCVRPTW is successfully converted into an SDCVRPTW. An algorithmic model is then applied to solve the transformed SDCVRPTW, generating a set of vehicle service routes. These routes all originate from the virtual depot, pass through designated service points, and satisfy all predefined constraints. As illustrated in Fig. 1, subgraph (a) depicts the original MDCVRPTW, while subgraph (b) shows the transformed SDCVRPTW. The final step involves the allocation of these routes to real depots. For each generated route, the optimal real depot is determined by the criterion that the sum of distances from the depot to both the start and end nodes of the route is minimized. Meanwhile, a constraint is imposed to ensure that the number of routes allocated to any real depot does not exceed the number of vehicles available at that depot. This allocation strategy ensures that each route is assigned to the real depot with the smallest associated distance cost, thereby minimizing the total cost aggregated across all routes.

Phase 1: Solve with Virtual Depot

MDP Formulation

The solution is constructed through a sequential decision process modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P_a, R_a), where S denotes the comprehensive state space, A stands for the complete action space, P_a represents the state transition probability, and R_a corresponds to the reward function. An agent iteratively selects the next flight to be served based on a learned policy π, generating a complete service sequence in a finite number of steps. Specifically, at each decision step t,

- *State*: The state at step t, denoted as $s_t \in S$, is a composite representation integrating both static and dynamic information. The static component includes graph embedding features, while the dynamic component encompasses the flight selected in the previous step and context-based embedding information.
- *Action*: The action $a_t \in A$ refers to the selection of the next flight to serve, which is determined by the policy $\pi(a_t|s_t)$ under the current operational state. Here, the action space A is equivalent to the node set $\{0, 1, ..., n\}$, where the node 0 is designated as the virtual depot and nodes 1 to n correspond to individual flight indices.

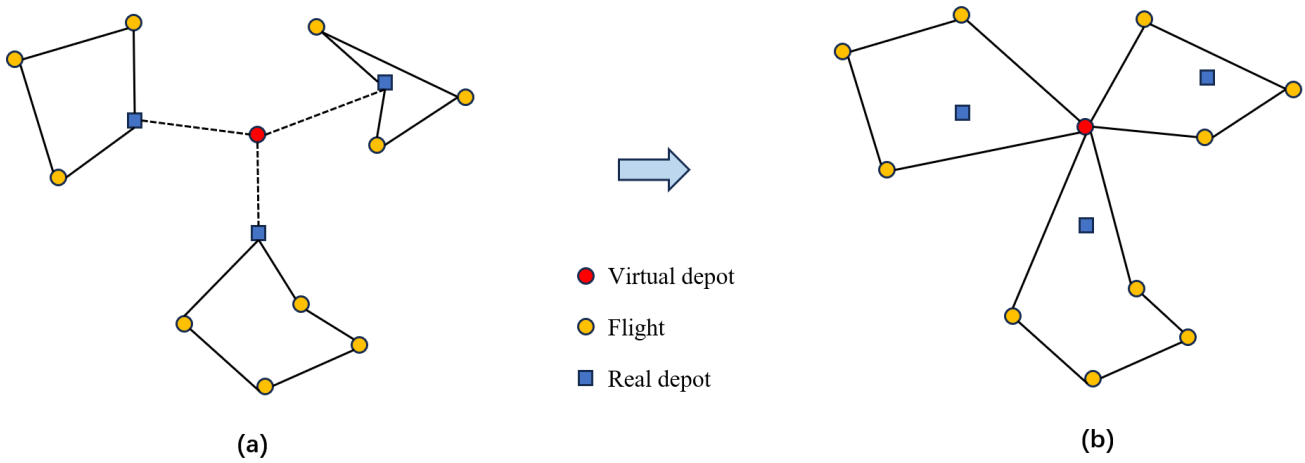

Figure 1. Illustration for the transformation process of the MDCVRPTW into a SDCVRPTW

- *Transition*: The transition probability $P_a = P(s_{t+1}|s_t, a_t)$ quantifies the probability of transitioning from the current state s_t to the next state s_{t+1} after executing action a_t. Notably, the state transition in this study is deterministic given a specific action, resulting in a transition probability of $P_a = 1$ for the corresponding state transition path.
- *Reward*: Once all flights are processed within T decision steps, a complete routing solution (i.e., a trajectory) is formed, expressed as $\tau = (a_1, a_2, ..., a_T)$. The total reward of the entire decision process is defined as the negative value of the trajectory length, i.e., $R_a = -L(\tau)$, where $L(\tau)$ denotes the total length of the constructed trajectory.

Model structure

Policy Parameterization. Given a instance x for AGS, the probability of a solution $\pi_\theta(\tau|x)$ is parameterized by θ and formulated as follows,

$$\pi_\theta(\tau|x) = \prod_{t=1}^{T} \pi_\theta(a_t|x, a_{1:t-1}), \tag{8}$$

where $a_{1:t-1}$ denotes the current partial solution (i.e., the trajectory) that has been constructed prior to the t-th iteration. A key aspect of the policy design is that at each decision step t, the policy π_θ must maintain Markovian dependence which means its output should depend solely on the last selected node and the initial node, without conditioning on the complete history of previously selected nodes. To learn the policy π_θ, we develop an encoder-decoder framework built upon the Transformer-based architecture which is show in Fig.2.

Encoder. The encoder is composed of N stacked multi-head attention (MHA) layers. Each node i (where $i = 0, \ldots, n$) is associated with a feature vector $\mathbf{f_i} = [d_i, tws_i, twe_i]$, representing its demand d_i, time window start tws_i, and end twe_i. These raw features are first projected into a higher-dimensional embedding space via a linear layer to obtain the initial node embeddings $\mathbf{h_i}$. Each MHA layer incorporates skip connections and is followed by instance normalization (IN) to stabilize and accelerate training.

$$\hat{\mathbf{h}}_\mathbf{i}^{(\mathbf{l})} = \mathrm{IN}\left(\mathbf{h}_\mathbf{i}^{(\mathbf{l-1})} + \mathrm{MHA}\left(\mathbf{h}_\mathbf{0}^{(\mathbf{l-1})}, \mathbf{h}_\mathbf{1}^{(\mathbf{l-1})}, \ldots, \mathbf{h}_\mathbf{n}^{(\mathbf{l-1})}\right)\right) \tag{9}$$

$$\mathbf{h}_\mathbf{i}^{(\mathbf{l})} = \mathrm{IN}\left(\hat{\mathbf{h}}_\mathbf{i} + \mathrm{FF}\left(\hat{\mathbf{h}}_\mathbf{i}\right)\right) \tag{10}$$

where l and $l-1$ denote the current and previous MHA layers, respectively. The feed-forward (FF) network comprises a hidden sublayer equipped with ReLU activation functions. The encoding procedure yields the final node embeddings $\mathbf{h}_\mathbf{i}^{(\mathbf{N})}$. Importantly, this encoding is executed a single time, and the resulting static embeddings are cached and reused across all subsequent decoding steps.

Decoder. The decoder constructs a solution in a sequential manner. At each step t, the input of the decoder comprises two components: the node embeddings $\mathbf{H_g} = \left(\mathbf{h}_\mathbf{0}^{(\mathbf{N})}, \mathbf{h}_\mathbf{1}^{(\mathbf{N})}, \ldots, \mathbf{h}_\mathbf{n}^{(\mathbf{N})}\right)$ and the dynamic context embeddings $\mathbf{h_c} = [\bar{\mathbf{h}}^{(\mathbf{N})}, \mathbf{h}_\mathbf{t-1}^{(\mathbf{N})}, Q_t]$,

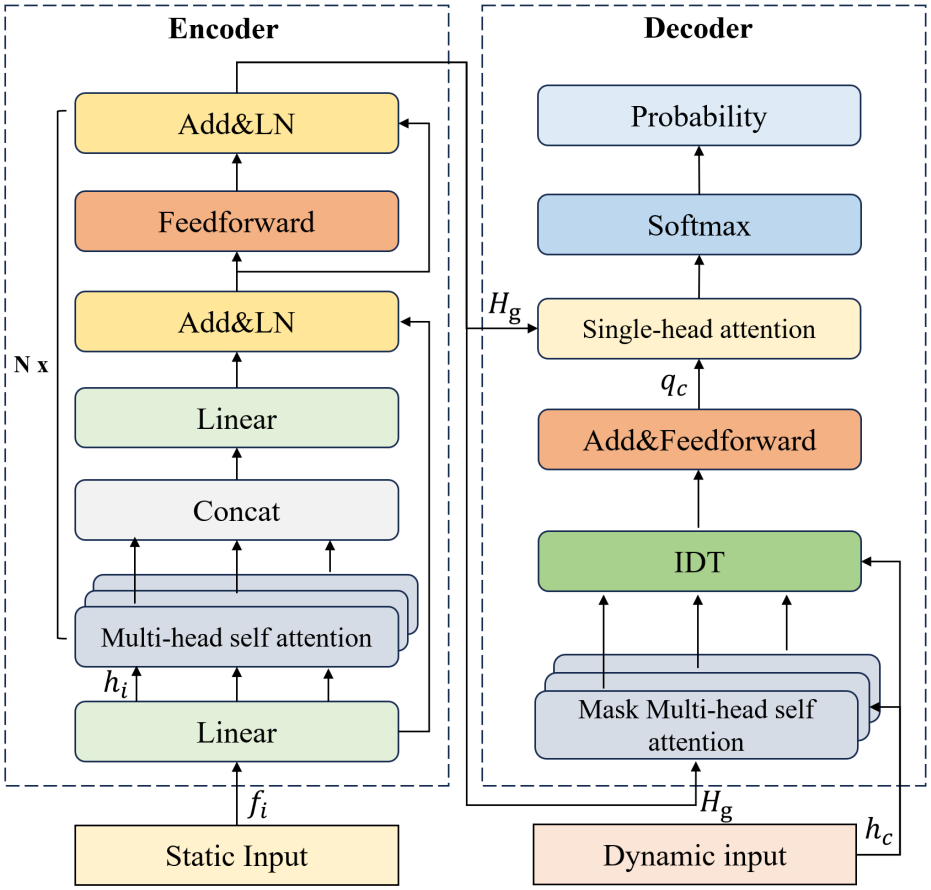

Figure 2. The architecture of our model

where $\bar{\mathbf{h}}^{(\mathbf{N})} = \frac{1}{n+1}\sum_{i=0}^{n}\mathbf{h_i^{(N)}}$ denotes the mean pooling of the node embeddings, $\mathbf{h_{t-1}^{(N)}}$ is the embedding of the last selected node, and Q_t refers to the remaining capacity of the current vehicle. We set the initial condition $a_0 = 0$ and initialize all vehicle trajectories to originate from the virtual depot node 0, with each individual trajectory being associated with a single vehicle. Note that Q_t is reinitialized when a vehicle select the virtual depot 0, practically meaning a new vehicle starts from the virtual depot to service unvisited flights. The decoder is composed of one MHA layer and one clipped Single-Head Attention (SHA) layer. Detailed structural specifications of the MHA and SHA layers can be referred to in the work by Kool et al[27].

$$\hat{\mathbf{h}}_\mathbf{c} = \mathrm{MHA}(\mathbf{h_c}, \mathbf{H_g}, \mathbf{H_g}) + \mathrm{IDT}(\mathbf{h_c}) \tag{11}$$

$$\mathrm{IDT}(\mathbf{h_c}) = \bar{\mathbf{h}}^{(\mathbf{N})} + \mathbf{h_{t-1}^{(N)}} + \mathbf{W}^{\mathrm{IDT}} Q_t \tag{12}$$

$$\mathbf{q_c} = \hat{\mathbf{h}}_\mathbf{c} + \mathrm{FF}(\hat{\mathbf{h}}_\mathbf{c}) \tag{13}$$

$$\mathbf{z_0}, \mathbf{z_1}, \ldots, \mathbf{z_n} = \mathrm{SHA}(\mathbf{q_c}, \mathbf{H_g}, \mathbf{H_g}) \tag{14}$$

where $\mathbf{W}^{\mathrm{IDT}}$ is a learnable parameter that projects Q_t to the same dimension as $\bar{\mathbf{h}}^{(\mathbf{N})}$ and $\mathbf{h_{t-1}^{(N)}}$, $\mathrm{IDT}(\cdot)$ denotes the identity mapping function and $\mathbf{q_c}$ is a powerful query that both derived from the approach introduced by Huang et al.[31]. The output

embedding $\mathbf{q_c}$ serves as the input to the SHA layer and the SHA layer then computes the probability p_i of selecting each candidate node as the next service node via the softmax function, which is formulated as:

$$p_i = \frac{e^{\mathbf{z_i}}}{\sum_j e^{\mathbf{z_j}}} \tag{15}$$

At each decision step, we enforce feasibility constraints by masking invalid nodes from the selection space. Specifically, we maintain a masking vector $\mathbf{m}_t$ whose elements correspond to node eligibility. The entries for prohibited nodes are assigned a value of $-\infty$, effectively excluding them from the attentional weighting prior to the softmax operation. In addition to masking nodes already selected in earlier steps, the masking mechanism incorporates constraint-specific infeasibility criteria. For instance, nodes that violate time window constraints or whose demand exceeds the remaining capacity of the current vehicle should be masked.

Policy Training. The policy parameterized by the Transformer-based architecture is trained using the REINFORCE algorithm enhanced with a shared baseline following Kwon et al.[28]. The objective of the training process is to minimize the expected cost of the solutions generated by the policy, mathematically formulated as:

$$\mathscr{L}(\theta|x) = \mathbb{E}_{\pi_\theta(\tau|x)}[L(\tau)] \tag{16}$$

where $L(\tau)$ represents a cost function that quantifies the length of the trajectory τ constructed by the policy. For the purpose of optimizing the policy parameters, gradient backpropagation is performed using the action sequence a output by the policy network. Concretely, the parameter update rule for the policy network is defined as follows:

$$\nabla\mathscr{L}(\theta|x) = \frac{1}{N}\sum_{i=1}^{N}(R(\tau^i) - b^i(x))\nabla_\theta \log p_\theta(\tau^i|x) \tag{17}$$

where N denotes the count of starting nodes a_1^i. It should be noted that the reward values $\{R(\tau^1), R(\tau^2), \ldots, R(\tau^N)\}$ are computed only after the complete construction of all corresponding trajectories $\{\tau^1, \tau^2, \ldots, \tau^N\}$. The term $p_\theta\left(\tau^i|x\right) = \prod_{t=2}^{T} p_\theta\left(a_t^i|x, a_{1:t-1}^i\right)$ corresponds to the product of the selection probabilities generated by the decoder at each step of trajectory construction. Additionally, the shared baseline is defined as $b^i(x) = \frac{1}{N}\sum_{j=1}^{N} R\left(\tau^j\right)$, which is the average reward across all sampled trajectories.

The full procedure for the training process is detailed in Algorithm 1, where L_i represents the solution cost (equivalent to the trajectory length) of the i-th instance when processed by the policy currently being trained, while $\mathscr{B}_i$ denotes the corresponding solution cost for the same instance using the optimally preserved model checkpoint. In line with this definition, the collections $\{L_i\}_{i=1}^{B}$ and $\{\mathscr{B}_i\}_{i=1}^{B}$ aggregate the solution costs generated by the two models (the current policy and the best saved model) across all training episodes and instances. Subsequent to the completion of each training epoch, both the in-training model and the best-performing checkpoint are subjected to greedy evaluation on a pre-separated validation dataset X_{val}. Their performance is then compared using a paired t-test on the validation results. If the current model shows a statistically significant improvement over the saved checkpoint, it replaces the previous best model as the new checkpoint.

Phase 2: Route Assignment

After obtaining the solution to the SDCVRPTW in Phase 1, a heuristic algorithm is designed to assign the route of each vehicle in the solution to a real depot, deriving the solution to the MDCVRPTW. Specifically, the problem can be formulated as follows: Given k vehicle service routes (each consisting of an origin and a destination),m depots with n vehicles each (where $m \times n \geq k$), the objective is to allocate each service route to a specific depot such that the total distance from the depot to the origin and destination of each assigned service route is minimized. In this paper, a Simulated Annealing (SA) algorithm is proposed to address this route assignment problem which has strong global search capability and fast convergence speed, and its pseudocode is presented in Algorithm 2.

where *NeighborSearch* represents generating neighboring solutions X_{new} for the current solution $X_{current}$, such as random perturbations, swaps, insertions and etc, *metropolis* represents the metropolis acceptance criterion[39], which means that if $\Delta f \leq 0$, it is accepted directly $X_{current} = X_{new}$, otherwise it is accepted with probability $P = exp(-\Delta f/T)$. Through the above solving process of the SA algorithm, we assign the service route corresponding to each vehicle to a specific actual depot, ultimately obtaining the solution to the MDCVRPTW in AGS. Moreover, we also tried several other heuristic algorithms to solve the allocation problem, and the specific solution results and performance comparison are shown in Ablation Studies. Comprehensive experimental evaluations presented in the subsequent section will demonstrate that our proposed approach outperforms both classical optimization methods and state-of-the-art baselines with statistically significant advantages.

Algorithm 1: Policy Optimization with REINFORCE

Input: number of epochs E; number of iterations per epoch I; number of starting nodes per sample N; batch size B; significance α.

Initialize policy network parameter θ, $\theta^* \leftarrow \theta$
generate random validation set X^{val}
for epoch $= 1$ **to** E **do**
  for iter $= 1$ **to** I **do**
    generate random training instances $x_i \quad \forall i \in \{1,\ldots,B\}$
    $\{a_i^1, a_i^2, \ldots, a_i^N\} \leftarrow \text{SelectStartNodes}(x_i) \quad \forall i \in \{1,\ldots,B\}$
    $\tau_i^j \leftarrow \text{SampleRollout}(a_i^j, x_i, \pi_\theta) \quad \forall i \in \{1,\ldots,B\}, \forall j \in \{1,\ldots,N\}$
    $b_i \leftarrow \frac{1}{N}\sum_{j=1}^{N} R(\tau_i^j) \quad \forall i \in \{1,\ldots,B\}$
    obtain gradient $\frac{1}{B}\nabla\mathscr{L}$ using Eq. (17)
    $\theta \leftarrow \text{Adam}(\theta, \frac{1}{B}\nabla\mathscr{L})$
  end for
  $\{L_i\}_{i=1}^{|X^{\text{val}}|} \leftarrow \text{Greedy}(X^{\text{val}}, \theta)$
  $\{\mathscr{B}_i\}_{i=1}^{|X^{\text{val}}|} \leftarrow \text{Greedy}(X^{\text{val}}, \theta^*)$
  if *t-test*$(\{L_i\}_{i=1}^{|X^{\text{val}}|}, \{\mathscr{B}_i\}_{i=1}^{|X^{\text{val}}|}) < \alpha$ **then**
    $\theta^* \leftarrow \theta$
    generate a new random validation set X^{val}
  end if
end for

Algorithm 2: Route Assignment with SA

Input: Initial solution $X_{current}$, initial temp T, termination temp T_{min}, initial temp decay β, initial current best solution $X_{best} = x_0$, objective function $f(x)$, maximum number of non-improvement steps C

while $T > T_{min}$ and $C < 1000$ **do**
  $X_{new} = NeighborSearch(X_{current})$
  $\Delta f = f(X_{new}) - f(X_{current})$
  $X_{current} = metropolis(\Delta f)$
  if $f(X_{current}) < f(X_{best})$ **then**
    $X_{best} = X_{current}$
  end if
  $T = \beta \times T$
end while
return X_{best}, $f(X_{best})$

Experiments

We evaluated the proposed method on AGS instances with multiple problem scales, especially in large-scale AGS. Comparative experiments were conducted against open-source solvers, classical meta-heuristic algorithms, and state-of-the-art neural learning-based methods to demonstrate the performance advantages of our method. In addition, we compared the training curves of various neural learning-based methods to illustrate the convergences during the training process. We further carried out ablation experiments to verify the contribution of the key components in the proposed method. Finally, we conducted tests in real-world scenarios to prove that the proposed method has the potential for practical deployment and application.

Instance Generation. Our real-world experimental study was conducted based on a hub airport in Southwest China, which consists of 2 terminals, 234 apron stands, and 4 depots, with each depot equipped with 20 vehicles. The apron stands are fixed in the airport layout, and the pairwise distances between all positions are pre-calculated. We generated 4 scales of AGS instances, namely AGS100, AGS300, AGS500, and AGS1000, corresponding to 100, 300, 500, and 1000 flights respectively. In each instance, flights were randomly assigned to apron stands, and 4 types of flights were considered, with each flight randomly allocated to one type. Each type represents a distinct turnaround time (i.e., 30, 33, 40, and 45 minutes). The arrival times of flights were sampled based on the actual operational statistics of the hub airport, while the departure times were determined by adding the turnaround time to the arrival times, so as to ensure sufficient time for ground handling operations. We set vehicle

speed according to Wu et al[26]. In addition, we set the capacity of each vehicle to 50, 70, 90, and 140 for AGS100, AGS300, AGS500, and AGS1000 respectively. Before inputting the flight demands into the neural network, they were normalized according to the vehicle capacity, resulting in an initial capacity representation of 1. For each problem scale (i.e., AGS100, AGS300, AGS500, and AGS1000), the policy network was trained for 100 epochs, with 12,800 training instances generated per epoch. After the training of each epoch, the model was evaluated on an independent validation set of 1,000 randomly generated instances of the same scale, in order to monitor the model performance and retain the optimal model. Finally, using different random seeds from those employed during training, an additional 1,000 instances were generated for each problem scale (i.e., AGS100, AGS300, AGS500, and AGS1000) for model testing.

Baselines. A comprehensive comparative analysis is carried out between our method and three distinct types of baseline methods, which are detailed as follows: (1) OR-Tools[40], a widely adopted classic solver for combinatorial optimization problems; (2) LKH3[41] and HGS[42], two well-established meta-heuristic algorithms; (3) AM[27], POMO[28], LEHD[13], and ReLD[31], four neural learning-based approaches.

Model Settings. The encoder comprises 5 layers, with the MHA module configured to have 8 heads. The embedding dimension is set to 128, and the hidden layer dimension is 512. The Adam optimizer is adopted for updating model parameters, using a constant learning rate of 10^{-4}. For the paired t-test, the significance level α is specified as 0.05. As for SA algorithm, the initial temperature T is set to 10,000, the termination temperature T_{min} is set to 0.01, the temp decay β is set to 0.9, and the maximum number of non-improvement steps is set to 1,000. All experimental trials were performed on a server with two Nvidia Quadro RTX 6000 GPUs and an Intel Xeon Gold 5220R CPU operating at 2.20GHz.

Overall comparison

A comparative analysis was conducted between the proposed neural method and the baseline method on the AGS100, AGS300, AGS500, and AGS1000 instances, with the average performance of objective value, primal gap and runtime on the testing set presented in Table 1, where our NCH4AGS indicates Neural Construction Heuristic For Airport Ground Scheduling. Note that the four neural learning-based methods AM, POMO, LEHD, and ReLD can only solve SDCVRPTW, so we will also use our two-stage strategy to solve MDCVRPTW for these methods. Specifically, the HGS algorithm demonstrates a robust capability to stably converge to optimal solutions, despite the high computational time cost. For open-source solvers, OR-Tools exhibits barely acceptable performance in handling small-scale instances (i.e., AGS100 and AGS300), however, as the problem scale gradually increases, both its solution quality and computational efficiency deteriorate notably, rendering it unfit for large-scale optimization tasks. In contrast, all neural network-based models outperform meta-heuristic algorithms in terms of solution efficiency, enabling the efficient generation of high-quality feasible solutions. Among these neural network models, our NCH4AGS achieves the superior performance in large-scale tasks (i.e., AGS500 and AGS1000), surpassing a range of existing neural network models. Notably, the greedy version of our proposed method incurs nearly no additional computational overhead, while the ReLD model shows marginally better performance than the proposed method in small-scale tasks (i.e., AGS100 and AGS300). Additionally, the aug8 version of our proposed method delivers the best overall solution quality, but the corresponding computation time increases significantly. Furthermore, the AM and POMO models perform well in small-scale problems but experience a significant decline in solution effectiveness with the expansion of problem scale, indicating that further improvements based on these models are necessary to enhance their adaptability to large-scale solving scenarios.

Table 1. Comparison with baselines.

	AGS100			AGS300			AGS500			AGS1000		
Methods	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
OR-Tools	132823.62	1.36%	12m	388221.57	7.16%	1.1h	631375.75	8.06%	2.5h	1173123.58	11.77%	4.2h
HGS	131028.87	*	35m	362289.22	*	1.5h	584251.91	*	3.6h	1049564.28	*	5h
LKH-3	132234.13	0.92%	1.1h	369283.64	1.93%	2.3h	596926.74	2.16%	4.6h	1087422.56	3.60%	7.4h
AM	133821.23	2.13%	3s	384512.45	6.13%	5s	646312.90	10.62%	8s	1242332.28	18.36%	13s
POMO(Aug8)	133124.62	1.59%	3s	371822.75	2.63%	**5s**	633821.74	8.48%	8s	1217413.72	15.99%	13s
LEHD	134235.23	2.44%	6s	376192.74	3.83%	11s	617828.26	5.74%	25s	1137821.23	8.40%	1.2m
ReLD	132923.51	1.44%	3s	369923.82	2.10%	5s	610123.26	4.42%	8s	1106211.63	5.39%	14s
NCH4AGS(Greedy)	133088.82	1.57%	4s	370542.63	2.27%	6s	607190.28	3.92%	9s	1098414.57	4.65%	15s
NCH4AGS(Aug8)	**132419.90**	**1.06%**	7s	**367864.12**	**1.53%**	15s	**598216.86**	**2.39%**	46s	**1086624.28**	**3.53%**	3.6m

Training Curve

As summarized from the experimental training curves in Fig.3, notable discrepancies in training stability and convergence behavior are observed among different models when tackling large-scale problems. Specifically, models explicitly designed for large-scale scenarios, including NCH4AGS, ReLD, and LEHD, exhibit significantly more stable training processes which means their loss values drop rapidly in the initial epochs and gradually tend to converge in subsequent training stages. In contrast, models not tailored for large-scale tasks (i.e., AM, POMO) display more volatile training trajectories and slower convergence rates when confronted with such problems. Among the large-scale-oriented models, both the proposed method and ReLD demonstrate smoother overall training curves and superior convergence outcomes, with the proposed method ultimately achieving the optimal convergence performance. Additionally, benefiting from the integration of symmetric data augmentation, the POMO model maintains a relatively more stable training process than the AM model when addressing large-scale problems, highlighting the positive effect of symmetric data augmentation on enhancing training stability for large-scale problems.

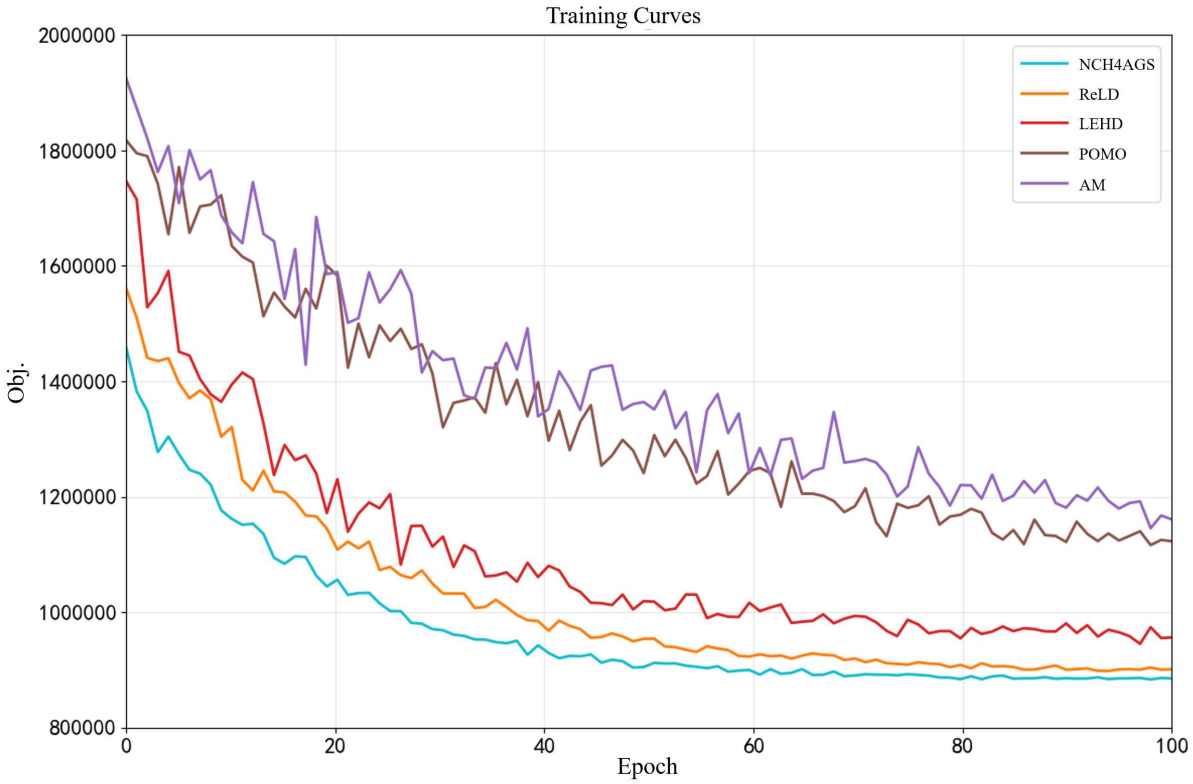

Figure 3. Training curves for neural learning-based methods on AGS1000.

Ablation Studies

Additionally, we performed ablation studies on the core components of our proposed framework to validate their individual contributions. The experimental results are reported in Table 2, where * represents the standard model configuration of this paper. Regarding the number of encoder layers, the configuration with 5 layers achieves the optimal balance between solution performance and time cost. Specifically, reducing the number of encoder layers leads to a decrease in computational time but concurrently impairs the solution quality. When the number of encoder layers is increased to 7, a marginal improvement in solution quality is observed for small-scale problems (i.e., AGS100 and AGS300), however, as the problem scale expands (i.e., AGS500 and AGS1000), the solution quality deteriorates. When the number of encoder layers is further increased to 9, both the solution quality and computational efficiency are compromised. In terms of normalization methods, the model adopting Batch Normalization (BN) exhibits inferior solution quality compared to the one using IN, though it incurs a slight reduction in computational time. This observation indicates that while BN contributes to more stable model training, IN is more suitable for large-scale MDCVRPTW, as it better facilitates the generation of high-quality solutions.

For the comparative experiments on heuristic algorithms for vehicle routing assignment in the second phase, in addition to the SA algorithm designed in this paper, the experiment also involves the Genetic Algorithm (GA), Particle Swarm Optimization (PSO) algorithm and Ant Colony Optimization (ACO) algorithm. Regarding the hyperparameter settings, the population size

Table 2. Ablation study

Method	AGS100		AGS300		AGS500		AGS1000	
	Gap	time	Gap	time	Gap	time	Gap	time
NCH4AGS-3Layer	2.16%	3s	2.64%	5s	5.42%	8s	6.18%	13s
NCH4AGS-7Layer	**1.38**%	4s	**2.19**%	6s	4.17%	10s	5.32%	16s
NCH4AGS-9Layer	2.02%	4s	2.54%	7s	4.78%	11s	5.91%	18s
NCH4AGS-BN	2.10%	4s	2.84%	5s	4.60%	8s	5.82%	14s
NCH4AGS-5Layer-IN*	1.57%	4s	2.27%	6s	**3.92**%	9s	**4.65**%	15s

of GA is set to 100, with the crossover probability of 0.7 and the mutation probability of 0.3. The population size of PSO is configured as 100, where the inertia weight is 0.6, the individual learning factor is 1.7 and the global learning factor is 1.8. As for ACO, the number of ants is 100, the pheromone evaporation coefficient is 0.2 and the pheromone intensity is set to 60. Detailed experimental results are presented in Table 3 which indicate that among the heuristic algorithms, the SA algorithm achieves the best performance while maintaining a relatively low solution time. The PSO algorithm exhibits the minimum solution time, however, it is prone to falling into local optima, resulting in only moderate performance. In contrast, both the GA and ACO algorithm demonstrate average performance in terms of both solution quality and time consumption, with the ACO algorithm showing suboptimal performance second only to the SA algorithm but requiring a longer solution time.

Table 3. Comparison of heuristic algorithms

Method	AGS100		AGS300		AGS500		AGS1000	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time
NCH4AGS+GA	1.66%	5s	**2.08**%	8s	4.79%	10s	5.82%	17s
NCH4AGS+PSO	2.92%	4s	3.24%	5s	6.13%	7s	6.75%	12s
NCH4AGS+ACO	**1.35**%	7s	2.66%	9s	4.07%	13s	5.23%	24s
NCH4AGS+SA	1.57%	4s	2.27%	6s	**3.92**%	9s	**4.65**%	15s

Real-world AGS Evaluation

In addition, experiments were conducted on the vehicle scheduling operations for AGS under real-world scenarios of the hub airport in Southwest China, covering three types of ground service tasks, including luggage, refueling, and catering. Specifically, real operational data of ground service tasks at the hub airport over a 30-day period were collected for testing. The problem scale ranged from the minimum to the maximum value within the 30 days, and the experimental results reported the 30-day average values, including the objective value, solution time, and the number of vehicles in use, as presented in Table 4. The proposed method exhibits excellent performance in real-world scenarios, with short solution time and stable solving effectiveness, and the quality of the obtained solutions approximately meets the requirements of ground service tasks. The number of vehicles used conforms to the demands of practical scenarios, and the number of vehicles allocated to each type of ground service task is lower than the total number of available vehicles, which indicates that the proposed method has the potential for practical deployment and application.

Table 4. Performance on real-world AGS scenario.

Task	Problem type	Problem size	Obj.	Time	Vehicle count
Luggage	MDCVRPTW	810-892	842621.26	14.4s	41.2
Refueling	MDCVRPTW	923-967	962735.37	14.9s	39.5
Catering	MDCVRPTW	989-1085	1123547.52	16.6s	56.8

Conclusion

This paper proposes a RL-based neural construction heuristic algorithm, which is capable of effectively solving the large-scale MDCVRPTW in AGS. Specifically, we decompose the entire solution process into two phases. In the first phase, the MDCVRPTW is transformed into a SDCVRPTW via a virtual depot, and solved based on the Transformer architecture. In the second phase, a heuristic algorithm is designed to allocate the obtained solutions according to the distance between vehicle routes and depots, assigning all vehicle routes to corresponding depots and deriving the final solution to the MDCVRPTW. Experiments conducted on the real-world scenario of a hub airport in Southwest China demonstrate that the proposed method outperforms existing open-source solvers, meta-heuristic algorithms, and state-of-the-art neural learning-based methods, which verifies the effectiveness of our method in addressing large-scale MDCVRPTW.

For future work, we intend to extend the model from single-task scheduling to multi-task joint scheduling in AGS scenarios, with the aim of tackling large-scale scheduling problems. Since multiple tasks are interrelated and coupled with each other, both the model complexity and computational complexity will be significantly increased. Therefore, we plan to introduce Graph Convolutional Network to model the correlation between tasks and capture their features, enhancing the decision-making capability of the model.

Data availability

All data generation information has been elaborated in detail in the experimental section. According to reasonable requirements, the corresponding author may provide the dataset in this study.

Funding

This work has been supported by the Sichuan Province Key R&D Program Provincial Institute Provincial School Science and Technology Cooperation Project Grant No.2026YFHZ0195.

Author contributions statement

Yang Zhang: Writing - Original draft, Conceptualization, Methodology, Data curation, Article proofreading. Mingjie Tang: Writing - Original draft, Conceptualization, Methodology, Data curation, Article proofreading. Hua Cheng: Methodology, Conceptualization, Article proofreading. Jizhen Ma: Writing review & editing, Methodology, Article proofreading. Runhao Xu: Writing review&editing, Data curation, Article proofreading. Fei Xiang: Writing – review & editing, Article proofreading.

References

1. Zhao, P., Gao, W., Han, X., Luo, W. *et al.* Bi-objective collaborative scheduling optimization of airport ferry vehicle and tractor. *Int. J. Simul. Model.* **18**, 355–365 (2019).
2. Zhu, S., Sun, H. & Guo, X. Cooperative scheduling optimization for ground-handling vehicles by considering flights' uncertainty. *Comput. & Ind. Eng.* **169**, 108092 (2022).
3. Gurobi Optimization, L. *Gurobi Optimizer Reference Manual* (2022). Accessed: 2023-01-01.
4. Du, J. Y., Brunner, J. O. & Kolisch, R. Planning towing processes at airports more efficiently. *Transp. Res. Part E: Logist. Transp. Rev.* **70**, 293–304 (2014).
5. Feng, X., Zuo, H. & Sun, Q. Research on collaborative scheduling of aircraft ground service vehicles based on simple temporal network. In *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, 263–269 (IEEE, 2021).
6. Ho, S. C. & Leung, J. M. Solving a manpower scheduling problem for airline catering using metaheuristics. *Eur. journal operational research* **202**, 903–921 (2010).
7. Guo, W. *et al.* Scheduling for airport baggage transport vehicles based on diversity enhancement genetic algorithm. *Nat. Comput.* **19**, 663–672 (2020).
8. Wang, C., Cheng, P., Li, J. & Sun, W. Leader reward for pomo-based neural combinatorial optimization. *arXiv preprint arXiv:2405.13947* (2024).
9. Berto, F. *et al.* Rl4co: an extensive reinforcement learning for combinatorial optimization benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 5278–5289 (2025).
10. Liu, F. *et al.* Multi-task learning for routing problem with cross-problem zero-shot generalization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1898–1908 (2024).

11. Gao, C., Shang, H., Xue, K., Li, D. & Qian, C. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *arXiv preprint arXiv:2308.14104* (2023).

12. Manchanda, S., Michel, S., Drakulic, D. & Andreoli, J.-M. On the generalization of neural combinatorial optimization heuristics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 426–442 (Springer, 2022).

13. Luo, F., Lin, X., Liu, F., Zhang, Q. & Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Adv. Neural Inf. Process. Syst.* **36**, 8845–8864 (2023).

14. Du, J. Y., Brunner, J. O. & Kolisch, R. Planning towing processes at airports more efficiently. *Transp. Res. Part E: Logist. Transp. Rev.* **70**, 293–304 (2014).

15. Han, X., Zhao, P., Meng, Q., Yin, S. & Wan, D. Optimal scheduling of airport ferry vehicles based on capacity network. *Annals Oper. Res.* **295**, 163–182 (2020).

16. Norin, A., Granberg, T. A., Värbrand, P. & Yuan, D. Integrating optimization and simulation to gain more efficient airport logistics. In *Eighth USA/Europe air traffic management research and development seminar*, vol. 10 (2009).

17. Norin, A., Yuan, D., Granberg, T. A. & Värbrand, P. Scheduling de-icing vehicles within airport logistics: a heuristic algorithm and performance evaluation. *J. Oper. Res. Soc.* **63**, 1116–1125 (2012).

18. Guo, W. *et al.* Scheduling for airport baggage transport vehicles based on diversity enhancement genetic algorithm. *Nat. Comput.* **19**, 663–672 (2020).

19. Zampirolli, K. A. & Amaral, A. R. S. Simulated annealing and iterated local search approaches to the aircraft refueling problem. In *International Conference on Computational Science and Its Applications*, 422–438 (Springer, 2021).

20. Du, Y., Zhang, Q. & Chen, Q. Aco-ih: An improved ant colony optimization algorithm for airport ground service scheduling. In *2008 IEEE International Conference on Industrial Technology*, 1–6 (IEEE, 2008).

21. Feng, X., Zuo, H. & Sun, Q. Research on collaborative scheduling of aircraft ground service vehicles based on simple temporal network. In *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, 263–269 (IEEE, 2021).

22. Padrón, S., Guimarans, D., Ramos, J. J. & Fitouri-Trabelsi, S. A bi-objective approach for scheduling ground-handling vehicles in airports. *Comput. & Oper. Res.* **71**, 34–53 (2016).

23. Padrón, S. & Guimarans, D. An improved method for scheduling aircraft ground handling operations from a global perspective. *Asia-Pacific J. Oper. Res.* **36**, 1950020 (2019).

24. Liu, Y., Wu, J., Tang, J., Wang, W. & Wang, X. Scheduling optimisation of multi-type special vehicles in an airport. *Transp. B: Transp. Dyn.* **10**, 954–970 (2022).

25. Zhou, J. *et al.* Learning large neighborhood search for vehicle routing in airport ground handling. *IEEE Transactions on knowledge data engineering* **35**, 9769–9782 (2023).

26. Wu, Y. *et al.* Neural airport ground handling. *IEEE Transactions on Intell. Transp. Syst.* **24**, 15652–15666 (2023).

27. Kool, W., Van Hoof, H. & Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475* (2018).

28. Kwon, Y.-D. *et al.* Pomo: Policy optimization with multiple optima for reinforcement learning. *Adv. Neural Inf. Process. Syst.* **33**, 21188–21198 (2020).

29. Kim, M., Park, J. *et al.* Learning collaborative policies to solve np-hard routing problems. *Adv. Neural Inf. Process. Syst.* **34**, 10418–10430 (2021).

30. Xin, L., Song, W., Cao, Z. & Zhang, J. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 12042–12049 (2021).

31. Huang, Z., Zhou, J., Cao, Z. & Xu, Y. Rethinking light decoder-based solvers for vehicle routing problems. *arXiv preprint arXiv:2503.00753* (2025).

32. Hou, Q., Yang, J., Su, Y., Wang, X. & Deng, Y. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *The Eleventh International Conference on Learning Representations* (2023).

33. Morsidi, F. An intensification-enhanced adaptive hybrid memetic algorithm for the multi-depot vehicle routing problem with time windows. *Appl. Model. Simul.* **10**, 50–63 (2026).

34. Xiang, F. *et al.* A gcn-based adaptive large neighbourhood search for multi-depot vehicle routing problem in airport ground handling. In *2024 8th Asian Conference on Artificial Intelligence Technology (ACAIT)*, 61–67 (IEEE, 2024).

35. Gu, Z. *et al.* Applying artificial bee colony algorithm to the multidepot vehicle routing problem. *Software: Pract. Exp.* **52**, 756–771 (2022).

36. Tohidifard, M., Tavakkoli-Moghaddam, R., Navazi, F. & Partovi, M. A multi-depot home care routing problem with time windows and fuzzy demands solving by particle swarm optimization and genetic algorithm. *IFAC-papersonline* **51**, 358–363 (2018).

37. Xue, S. An adaptive ant colony algorithm for crowdsourcing multi-depot vehicle routing problem with time windows. *Sustain. Oper. Comput.* **4**, 62–75 (2023).

38. Dursun, Ö. O. & Özger, A. Multi-depot heterogeneous fleet vehicle routing problem with time windows: Airline and roadway integrated routing. *Int. J. Ind. Eng. Comput.* **13**, 435–456 (2022).

39. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. *The journal chemical physics* **21**, 1087–1092 (1953).

40. Didier, F. *et al.* Or-tools' vehicle routing solver: a generic constraint-programming solver with heuristic search for routing problems (2023).

41. Helsgaun, K. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde Univ.* **12**, 966–980 (2017).

42. Vidal, T. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Comput. & Oper. Res.* **140**, 105643 (2022).