An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.15, No.4, pp.738-749 (2025)
https://doi.org/10.36922/IJOCTA025220109

RESEARCH ARTICLE

FastLoader: Leveraging large language models to accelerate cargo loading optimization with numerous loading constraints

Yunlai Cheng[1†], Zheng Chen[2†], Siqi Du[1], Meng Yu[2], Dongzhou Zhao[2], Xinran Li[2], Yue Han[2], Zhe Ouyang[2], Yinglong Wang[3], Jing Chen[3], Ying Guo[3], Chi Harold Liu[1], and Rui Han[1*]

[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2]TravelSky Technology Limited, Beijing, China
[3]Shandong Computer Science Center (National Supercomputer Center in Jinan)
and Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong, China
chenzh@travelsky.com.cn, 3120230948@bit.edu.cn, siqiqia@sina.com, myu@travelsky.com.cn, dzhzhao@travelsky.com.cn, lixinran@travelsky.com.cn, yuehan@travelsky.com.cn, ouyangzhe@travelsky.com.cn, wangyinglong@qlu.edu.cn, jingchen94@163.com, guoying@sdas.org, chiliu@bit.edu.cn, hanrui@bit.edu.cn

ARTICLE INFO

Article History:
Received: May 31, 2025
1st revised: July 24, 2025
2nd revised: September 2, 2025
Accepted: September 3, 2025
Published Online: September 19, 2025

Keywords:
Cargo loading
Combinatorial optimization
Optimization acceleration
Loading constraints
Large language model

AMS Classification 2010:
26A33; 34A08; 35H15; 34K50
47H10; 60H10

ABSTRACT

With the unquestionable commercial success of air cargo transportation, cargo loading is a crucial step that selects the optimal placement solution for a given aircraft hold and a set of cargoes. This combinatorial optimization promotes airlines' revenue (e.g., minimizing fuel consumption) with the encoded constraints in the solution space. In practical scenarios, cargo loading includes dozens of loading constraints (e.g., isolation of dangerous cargoes). However, existing techniques either over-simplify such constraints due to the expensive manual modeling in combinatorial optimization, or suffer from a time-consuming optimization process due to the large search space in heuristic search. In this paper, we present FastLoader, an optimization acceleration approach that employs large language models (LLMs) to distinguish critical structural patterns in the simulated cargo loading data while still scaling to numerous loading constraints in real scenarios. FastLoader's key design features are the following: (i) a cargo loading constructor, which converts the information of both cargo types and loading constraints into pre-defined data structures, thus avoiding manual modeling and improving solution accuracy; (ii) a cargo loading solver and a search space reducer, which work together to effectively reduce search space and accelerate the optimization process. We evaluate the proposed approach using a list of practical scenarios from industry transportation systems, and the results show the followin: FastLoader improves accuracy by 10% compared to combinatorial optimization, and reduces the optimization time by 90% with 1.5% accuracy losses compared to heuristic search.

1. Introduction

The rapid development of air cargo transportation[1,2] makes the air cargo loading a promising research field. As shown in Figure 1, air cargo loading is an industrial process of loading air cargoes (e.g., unit loading devices, bulk cargo, and special goods) onto flights (e.g., wide-body and narrow-body aircraft cargo holds) in three steps. The first step is to process and prepare cargoes for transformation. The second step

[†] *These authors contributed equally to this work.*
*Corresponding Authors

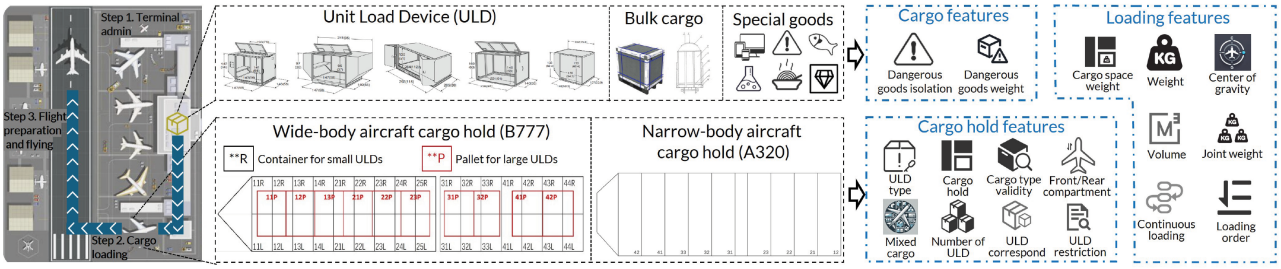

Figure 1. Air cargo loading:unit load devices, bulk cargo, and special goods into wide/narrow-body aircraft.

Table 1. Description of constraints in air cargo loading scenario

Constraint		Description
Cargo features	Special cargo space weight constraint	This constraint defines the maximum allowable weight of special cargo in the space
	Dangerous cargo isolation constraint	Any two special cargo loading locations need to maintain a specified distance
Aircraft cargo hold features	Unit Load Device (ULD) correspondence constraint	Get the corresponding relationship of cargo types and verify each piece of cargo data
	ULD Type Restriction Rules	If the container type in the loading data is not one of the ones defined in ULD Type, the check fails
	ULD type and ULD number constraint	If the container type does not correspond to the container serial number, the verification fails.
	Cargo hold availability constraint	Before loading, check whether the cargo hold is available
	Mixed cargo space constraint	Check whether there is mixed loading in the cargo hold
	Number of ULD constraint	The quantity of ULD cannot exceed this specified value
	Front/Rear compartment constraint	Ensure weight in the front (FWD) and rear (AFT) compartments does not exceed the defined limits.
	Cargo Type validity constraint	Check whether cargo type is valid and belongs to predefined cargo types.
Loading features	Weight constraint	Maximum load weight of the cargo hold
	Center of Gravity (CG) constraint	Ideal center-of-gravity range for airliner when zero fuel
	Volume constraint	The volume of cargo cannot exceed this specified value
	Joint weight constraint	Total load weight constraints for multiple cargo holds
	Cargo space weight constraint	This constraint defines the maximum weight limit for a cargo space.
	Continuous loading constraint	Some types of ULDs need to be loaded according to the load sequence
	Load order constraint	Goods must be loaded in the specified order

is to transfer and load cargoes onto the aircraft, and the third step is to prepare for the departure. The core goal of air cargo loading is to reduce the center-of-gravity shift after loading by optimizing, thereby saving fuel consumption and increasing aircraft safety.

Since the 1980s, many approaches[3,4] to air cargo loading have been proposed, falling into the following two categories: combinatorial optimization[5,6] and heuristic search.[7–11] However, since the scenarios of these approaches are oversimplified, none of these approaches can achieve high performance under complex constraints. Oversimplify cargo loading scenarios with only three cargo types and four constraints. However, in practical loading scenarios, as shown in Figure 1, there exist 13 cargo types and 17 loading constraints. The limitations of these previous approaches are divided into two challenges.

First, the combinatorial optimization (§ 2.2) approaches achieve low accuracy under complex constraints. Most combinatorial optimization approaches only consider the cargo features and two loading features (weight constraint and center-of-gravity constraint) in the constraints. They sacrifice the size of the search space to achieve faster solution speed, but it results in an unsalvageable final solution. This is because under complex constraints, optimization problem modeling is difficult and requires high expertise in scenarios. Therefore, simple scenario modeling reduces the accuracy of the solution. The first challenge is *how to accurately model scenarios with numerous cargo types and complex constraints.*

Second, the heuristic search (§ 2.3) approaches consume a lot of computation time to calculate solutions under complex constraints. To solve the accuracy drop in combinatorial optimization, heuristic search approaches increase the accuracy of the solution by increasing the size of the search space. However, most heuristic search approaches only consider the cargo features and the loading features in the constraints. Failure to consider cargo hold constraints increases the number of infeasible solutions in the search space, and it leads to unacceptable searching time costs. Therefore, our second challenge is *how to effectively reduce the search time and improve the solution speed while reducing the loss of search accuracy.*

In this work, we propose FastLoader, a large language model (LLM)[12–18] based optimization method to address the difficulty in optimization problem modeling and the high time consuming under complex constraints. The core idea of FastLoader is the search space reducer driven by LLM, which effectively reduces the size of the search space and improves the solution speed. In the optimization problem modeling stage, FastLoader models cargo data and complex constraints to form cargo information and LLM engine. In the search space reduction stage, FastLoader utilizes modeling data to quickly identify infeasible solutions in the search space and thus simplifies the

search space. The main contributions of this work are as follows.

- **Optimization problem modeling.** To address the first challenge, FastLoader introduces cargo loading constructor (§ 3.2) that builds a data structure of cargo information and complex constraints, which reduces the requirement of expertise in the modeling process and provides data input for the search space reduction stage.
- **Search space reduction.** To address the second challenge, FastLoader introduces a search-space reduction stage that augments conventional heuristic optimization driven by LLM. Concretely, we construct a cargo-loading solver (§ 3.3), which enumerates candidate load plans from the raw cargo manifest and engineered feature set. We also design a search space reducer (§ 3.4), which scores the resulting solutions and prunes those that violate key operational constraints extracted from the LLM engine.

Summary of experimental results. We evaluate FastLoader on four common aircraft types, and we conduct the evaluation using seven baselines across five LLMs. The results show the following: (1) FastLoader achieves the best performance compared to the seven baselines with only 1.5% accuracy loss; (2) in the comparison of time consuming between different heuristic search baselines and FastLoader, FastLoader reduces the time costs by 90% on average; (3) FastLoader achieves stable performance across 5 LLMs, which proves FastLoader is applicable to multiple LLMs.

2. Background & related work

In the cargo loading-optimization scenario(§ 2.1), existing methods can be divided into the following two categories: (1) combinatorial optimization(§ 2.2) methods are designed to solve cargo loading optimization under simplified constraints; (2) heuristic search (§ 2.3) methods are designed to solve cargo loading optimization under complex constraints.

2.1. Background

In typical air cargo loading scenario, cargo types fall into two broad categories: Unit Load Devices (ULDs) are large cargo containers suitable for wide-body aircraft (e.g., B777, B787). Bulk cargo comprises loose cartons, mailbags, live animals, and other items placed directly into the bin-shaped holds of narrow-body aircraft(e.g., B737, A320). As shown in Table 1, there are 17 complex constraints in our scenario, and they are divided into three categories including cargo features, cargo hold features, and loading features.

2.2. Combinatorial optimization methods

Combinatorial optimization is a mathematical optimization algorithm where each possible cargo position is explored under the simple constraints. The aim is to minimize center-of-gravity variation and satisfy all constraints. Existing research can be divided into linear programming such as COM,[6,19–22] nonlinear programming such as IOM,[5,23,24] and mixed-integer linear programming frameworks such as MLIP.[25–28]

COM[6] presents a joint integer linear programming model that solves the discretization of cargo under simple constraints. The model maximizes loading capacity and minimizes the center-of-gravity deviation. It performs well when constraints are simple but fails under complex constraints. Thus, it cannot deal with the cargo loading case investigated in this study. IOM[5] proposes an algorithm for the NP-hard cargo loading problem. The algorithm splits the task into four sub-problems: aircraft configuration, pallet assembly, air-cargo palletization, and center-of-gravity balancing. It integrates path optimization to maximize transport benefit. However, the four sub-problems do not cover all constraints in our scenario, so the method performs poorly when constraints become complex. MLIP[25] introduces a mixed-integer programming model that optimizes cargo center-of-gravity and inertia moment to improve loading efficiency and reduce fuel burn. Using real data constraints, the method greatly shortens search time. However, its data modeling fails to cope with highly constrained scenes. When constraints increase, MLIP cannot always find feasible solutions. MLIP-ACLPDD[28] employs a similar mixed-integer model. Its objective minimizes fuel consumption and loading cost. The method still shows low accuracy under complex constraints.

In conclusion, when applied to complex constraints scenario, existing combinatorial optimization methods lead to the following two main drawbacks: (1) complex constraint scenario requires people with professional knowledge to model, and the scenario modeling is difficult; (2) combinatorial optimization methods have difficulty finding the optimal solution within the modeling of complex constraints. As a result, the accuracy of combinatorial optimization methods is limited by the complex constraints scenario. Heuristic search is designed to solve these two drawbacks when the constraints become complex.

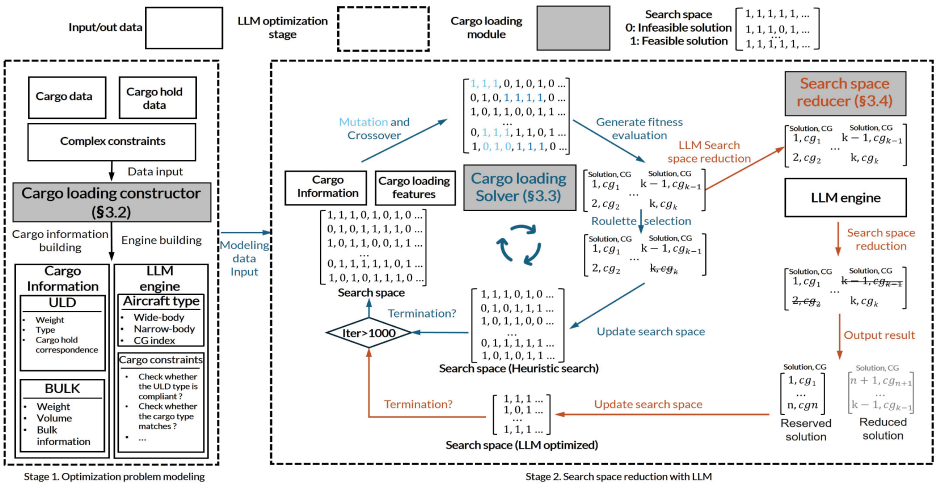

Figure 2. The architecture of FastLoader.

2.3. Heuristic search methods

In complex constraints scenario, heuristic search uses empirical rules and guidance to locate near-optimal solutions. It avoids exhaustive enumeration of all options. Such methods suit cargo loading optimization problems under complex constraints. Current studies are divided into genetic algorithms[10,29–33] and particle swarm optimization techniques.[8,9,11,34,35]

Genetic algorithms such as HGA[10] propose a hybrid genetic model for cargo loading optimization under complex constraints. The objective is to minimize the center-of-gravity shift and fuel burn. GA-normal[36] also employs a classical genetic algorithm for the same problem. DMOPSO[9] and PSO-normal[8] adopt particle swarm optimization. Their goals are to reduce fuel consumption and center-of-gravity deviation, respectively. They convert diverse constraints into penalty factors, which lowers modeling complexity. However, a large search space under complex constraints still yields a high computational cost. The papers[37,38] adopt the Maximum Expert Consensus Model (MECM) as the core framework. They incorporate satisfaction functions and uncertainty modeling for robust, preference-aligned group decision support. However, the search time is long under current multi-constraint and complex scenarios.

In conclusion, heuristic search continuously updates the optimal solution in the search space. It solves the drawbacks of combinatorial optimization in complex constraints scenario. However, in order to achieve higher solution accuracy, heuristic search needs to repeatedly update and iterate the search space, which greatly increases the time cost of the solution.

3. Method

3.1. Overview

FastLoader is designed based on two features of cargo loading optimization: (1) modeling the cargo loading scenario under complex constraints is difficult and requires a high degree of expertise; (2) the search space in heuristic search contains a large number of infeasible solutions,[10,29] and the time costs of finding solutions in the search space increases when the number of constraints in the cargo loading scenario increases. That is, simple modeling methods produce a larger search space, but in our scenario, feasible solutions in the search space only account for 40% of the search space. The remaining 60% of infeasible solutions negatively affect the accuracy and speed of the existing methods.

Figure 2 illustrates the architecture of FastLoader, which is split into two stages and three modules: optimization problem modeling (cargo loading constructor), search space reduction (classic cargo loading solver and search space reducer). We design optimization problem modeling and search space reduction to support cargo loading optimization under complex constraints. The goal of FastLoader is to reduce solution time costs while maintaining high accuracy.

First, the optimization problem modeling in FastLoader is able to clearly construct the basic information of cargoes and the modeling information required for the large language model (LLM). At the stage of optimization problem modeling, cargo loading constructor (§ 3.2) takes cargo data, cargo hold data, and loading constraints as input. FastLoader also develops two data structures as the output constructor to support problem solving and search space reduction. The cargo information contains two types of cargo attributes.

Table 2. Experimental results of different algorithms for air cargo loading on wide-body aircraft.

Algorithm		B777			B787		
		MAC(%)↓	INDEX(%)↓	TIME(s) ↓	MAC(%) ↓	INDEX(%) ↓	TIME(s) ↓
FastLoader	Deepseek	24.30±0.82	10.00±6.51	2.14±0.20	24.32±0.72	10.00±5.43	2.17±0.18
	Llama	24.20±0.58	9.17±4.73	1.04±0.01	24.31±0.61	9.77±4.58	1.04±0.00
	Phi	24.22±0.64	9.33±5.05	1.36±0.01	24.12±0.41	8.20±2.82	1.39±0.05
	Qwen-1.5	24.38±0.85	10.68±6.84	1.87±0.10	24.10±0.48	**7.97±3.44**	1.68±0.35
	Qwen-3	24.14±0.62	**8.65±5.08**	1.76±0.49	24.19±0.72	8.79±5.86	1.70±0.46
Combinatorial optimization	COM	25.39±0.98	16.21±5.80	**0.02±0.01**	25.24±0.83	15.76±5.03	0.04±0.01
	IOM	25.30±0.96	15.84±5.64	0.03±0.01	25.16±0.80	15.61±5.03	**0.03±0.01**
	MLIP	25.25±0.97	15.75±5.78	0.05±0.01	25.17±0.84	15.58±5.20	0.12±0.01
Heuristic search	HGA	23.25±0.47	9.14±3.25	110.21±25.5	23.64±0.24	9.19±4.82	183.91±20.16
	GA-normal	**23.24±0.47**	9.26±3.11	118.30±17.98	23.68±0.24	9.56±4.36	185.54±20.95
	DMOPSO	23.36±0.38	9.40±4.10	135.21±26.36	23.56±0.31	9.25±3.78	269.48±28.15
	PSO-normal	23.39±0.69	9.30±3.47	160.19±24.63	**23.38±0.69**	9.21±4.78	275.49±20.24

Table 3. Experimental results of different algorithms for air cargo loading on wide-body aircraft.

Algorithm		B777			B787		
		MAC(%)↓	INDEX(%)↓	TIME(s) ↓	MAC(%) ↓	INDEX(%) ↓	TIME(s) ↓
FastLoader	Deepseek	24.30±0.82	10.00±6.51	2.14±0.20	24.32±0.72	10.00±5.43	2.17±0.18
	Llama	24.20±0.58	9.17±4.73	1.04±0.01	24.31±0.61	9.77±4.58	1.04±0.00
	Phi	24.22±0.64	9.33±5.05	1.36±0.01	24.12±0.41	8.20±2.82	1.39±0.05
	Qwen-1.5	24.38±0.85	10.68±6.84	1.87±0.10	24.10±0.48	**7.97±3.44**	1.68±0.35
	Qwen-3	24.14±0.62	**8.65±5.08**	1.76±0.49	24.19±0.72	8.79±5.86	1.70±0.46
Combinatorial optimization	COM	25.39±0.98	16.21±5.80	**0.02±0.01**	25.24±0.83	15.76±5.03	0.04±0.01
	IOM	25.30±0.96	15.84±5.64	0.03±0.01	25.16±0.80	15.61±5.03	**0.03±0.01**
	MLIP	25.25±0.97	15.75±5.78	0.05±0.01	25.17±0.84	15.58±5.20	0.12±0.01
Heuristic search	HGA	23.25±0.47	9.14±3.25	110.21±25.5	23.64±0.24	9.19±4.82	183.91±20.16
	GA-normal	**23.24±0.47**	9.26±3.11	118.30±17.98	23.68±0.24	9.56±4.36	185.54±20.95
	DMOPSO	23.36±0.38	9.40±4.10	135.21±26.36	23.56±0.31	9.25±3.78	269.48±28.15
	PSO-normal	23.39±0.69	9.30±3.47	160.19±24.63	**23.38±0.69**	9.21±4.78	275.49±20.24

ULD records the weight, cargo type and cargo hold correspondence between cargo and the cargo hold. Bulk records the weight, cargo volume, and the bulk information, such as mail, luggage, etc. The LLM engine consists of aircraft type and cargo constraints. The cargo constraints include all important constraint information for the cargo loading scenario.

Second, the search space reduction is based on the existing heuristic search methods. We design two modules, including a classic cargo loading solver and a search space reducer. The cargo loading solver (§ 3.3) generates its search space and takes cargo information and cargo loading features as input. Then, the solution is completed through classic heuristic search method. The search space reducer (§ 3.4) calculates the score of each solution in the search space. At the same time, driven by LLM, it uses the important constraint information contained in the LLM engine to simplify the search space.

3.2. Cargo loading constructor

Cargo loading constructor converts the cargo information and complex constraints into simplified modeling data and inputs them into the next stage. There are two main steps in the cargo loading constructor, including cargo information building and engine building.

Cargo information building. In this step, we design a dictionary data structure to record the information of two main cargo types: Unit Load Device (ULD) and Bulk. The raw manifest is transformed into a high-resolution description of every freight item so that optimization algorithms can consume the data with no further cleaning. (1) Each record is first classified as either a ULD or bulk cargo. It is then enriched with all operational attributes: mass, ULD contour code (or bulk volume), special-handling tags such as "dangerous goods" or "perishables," and any stated stowage preferences or prohibitions. (2) The constructor also references airline and aircraft documentation to enumerate which cargo hold positions are structurally or geometrically compatible with that piece. The output is a pair of dictionaries keyed by unique cargo IDs. (3) Items that violate a structural rule at this stage (e.g., a heavy ULD that exceeds every individual hold limit) are flagged immediately.

Engine building. The second step assembles an LLM engine that fuses aircraft cargo hold information and complex constraints into a single data structure. (1) Numeric constants are loaded from the trim-sheet database and stored in a structured map. The numeric constants include the longitudinal coordinates of every hold, individual weight capacities, and the center-of-gravity (CG). (2) Complex constraints are then appended as human-readable clauses: examples

include *"Dry-ice must not share a sealed compartment with live animals"* or *"Lithium batteries may only occupy ULD positions equipped with fire-suppression liners"*. Unlike the numeric constants, these clauses are deliberately retained in natural language so that an LLM performs fast, semantics-level validation before a candidate plan is ever scored numerically. (3) LLM engine binds the cargo dictionaries to the aircraft map, constructing an output that is processed by cargo loading solver. The same output is serialized into a concise prompt for the LLM so that complex constraints are processed effectively by the LLM.

3.3. Cargo loading solver

Cargo loading solver only performs iteration searching in classic heuristic search to solve the air cargo loading problem. Note that each (k-th) iteration has its own search space, which is defined as a matrix of solutions $S^{(k)}$:

$$S^{(k)} = \begin{bmatrix} S_{1,1}^{(k)} & S_{1,2}^{(k)} & S_{1,3}^{(k)} & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ S_{2,1}^{(k)} & S_{2,2}^{(k)} & S_{2,3}^{(k)} & \cdots \\ S_{N,1}^{(k)} & S_{N,2}^{(k)} & S_{N,3}^{(k)} & \cdots \end{bmatrix} \tag{1}$$

where $S_{i,j}^{(k)}$ represents one of the k-th iteration's solutions, N represents the length of the search space. Here, i and j are used to identify the position of a candidate solution in the two-dimensional search space grid. In each iteration, the cargo loading solver updates the solution in the search space to facilitate the search for the optimal solution.

In the search space, not all solutions satisfy all the complex constraints in the scenario. We define the solution that satisfies all constraints as a feasible solution and assign it a value of 1. Similarly, an infeasible solution is assigned a value of 0:

$$S_{i,j}^{(k)} \leftarrow \begin{cases} 1 & \text{feasible solution} \\ 0 & \text{infeasible solution.} \end{cases} \tag{2}$$

A single-point mutation operator is applied after crossover, randomly flipping selected genes in the binary loading matrix to inject new cargo hold positions. Roulette wheel selection then preserves fitter individuals by drawing each candidate with probability $p_i = cg_i / \sum_j cg_j$ where cg_i represents its center-of-gravity fitness. Even after this evolutionary update, the expanded search space still contains a high density of infeasible solutions, so convergence typically demands thousands of iterations and incurs substantial computational time.

3.4. Search space reducer

We design search space reducer to reduce the substantial time caused by cargo loading solver. In each iteration of search, the search space reducer accepts the search space for crossover and mutation and the corresponding scoring matrix:

$$S_{score}^{(k)} = \begin{bmatrix} S_{1,1}^{(k)} & cg_{1,1} \\ S_{1,2}^{(k)} & cd_{1,2} \\ \vdots & \vdots \\ S_{N,1}^{(k)} & cg_{N,1} \\ \vdots & \vdots \\ S_{N,N}^{(k)} & cg_{N,N} \end{bmatrix} \tag{3}$$

where the $cg_{i,j}$ represents the center-of-gravity fitness of solution $S_{i,j}^{(k)}$. Driven by the LLM, the search space reducer quickly classifies the current infeasible solutions and achieves the goal of significantly simplifying the search space:

$$S_{update}^{(k)} = \begin{bmatrix} S_{1,1}^{(k)} & S_{1,2}^{(k)} & \cdots \\ \vdots & \vdots & \vdots \\ S_{n,1}^{(k)} & S_{n,2}^{(k)} & \cdots \end{bmatrix} \tag{4}$$

where the length of the search space after simplification, n, is much smaller than N before simplification. The simplified search space $S_{update}^{(k)}$ is input into the cargo loading solver to participate in the next round of heuristic search iteration.

4. Evaluation

In this section, we evaluate the performance of FastLoader compared with state-of-the-art baselines. We conduct experiments from the following three perspectives: performance evaluation, runtime decomposition, and module ablation analysis. The experiment comprises seven baseline air cargo loading algorithms, five large language models (LLMs), four representative aircraft types, and an industrial cargo dataset containing 6.1 million loading records.

4.1. Basic setting

Testbed. We choose five representative LLMs to implement FastLoader. The five LLMs include: (1) DeepSeek-R1[39] is a 671 billion-parameter transformer decoder trained on about two trillion multilingual tokens; (2) Llama-2-70B[40] is Meta's 70-billion-parameter open-source foundation model with an optimized transformer backbone for general text generation; (3) Phi-3 (3.8 B)[41] is Microsoft's 3.8-billion-parameter compact model, tuned on curated and synthetic data to outperform larger peers on reasoning tasks; (4)

Qwen-1.5-72B[42] is Alibaba's 72-billion-parameter multilingual model that supports up to 32 k context tokens and excels at chat, code, and tool use; (5) Qwen-3-235B[43] is a 235-billion-parameter mixture-of-experts model that activates roughly 22 billion parameters per token to achieve high accuracy with lower compute overhead.

Dataset. The experiments rely on an air cargo loading dataset collected by China Aviation Information Co. from June 2024 to January 2025 at an airport in Beijing, China. The data were collected at the flight level, with each record corresponding to a specific flight's cargo loading information. After initial cleaning to remove erroneous entries, the dataset comprises over 6.1 million cargo records spanning more than 1000 aircraft types. For this study, we sample 200,000 records drawn from two wide-body aircraft (B777, B787) and two narrow-body aircraft (A320, B737).

Baseline. We evaluate four state-of-the-art heuristic search methods, including HGA,[10] GA-normal,[10] DMOPSO,[9] and PSO-normal[8] and three combinatorial optimization methods, including COM,[6] IOM[5] and MLIP.[25]

Metric Following the setting of,[44] we utilize the following three metrics:

- **%MAC:** The Mean Aerodynamic Chord (MAC) represents the chord of a hypothetical rectangular wing whose projected area equals that of the real wing, thereby maintaining the same aerodynamic and moment characteristics. Although exact computation is challenging, the MAC is often established via wind tunnel testing. As a key factor in aircraft load analysis, the %MAC is derived based on this mean aerodynamic chord. The MAC is calculated as

$$\%MAC = \frac{\frac{C\cdot(I-K)}{W} + RA - LEMAC}{MAC} \times 100 \quad (5)$$

  where C is the weight constant of the aircraft, I is the index value corresponding to the respective weight, K is the constant that is used to avoid the negative figures, W is the actual weight of the aircraft, RA is the reference index values, LEMAC is the horizontal distance in length units from reference datum to the location from the leading edge, and MAC is the length of mean aerodynamic chord in length units. The smaller %MAC represents the better performance of algorithms.
- **The change rate of index:** Index is another significant concept in aircraft cargo loading, it is calculated as

$$Index = \frac{W \cdot (BA - RA)}{C} + K \quad (6)$$

  where BA is the horizontal distance in units of length from the reference datum to the location. The smaller index change rate represents the better performance of algorithms.
- **Algorithm runtime:** Algorithm runtime refers to the measure of execution time. The lower algorithm runtime represents the better performance of algorithms.

4.2. Evaluation of loading performance

Hyperparameter setting. For the genetic algorithms (HGA and GA-normal), we set a population of 10,000, run 100 generations, and use a mutation rate of 0.7 with a crossover rate of 0.2. For the particle-swarm optimizations (DMOPSO and PSO-normal), we employ 2000 particles and 100 iterations. Our own method runs for 100 iterations on a search space of 100 individuals and sets the LLM temperature at 0.8, top_p 0.9, and a repetition-penalty of 1.2.

Comparison of loading accuracy. Table 2 and Table 3 show the comparison results of all eight methods and five testbeds. We have three key observations, which are following:

(1) *Generalization analysis of FastLoader.* Compared with the combinatorial optimization method, FasterLoader shows higher and more stable solution accuracy, with higher generalization performance. This is because the LLM fully understands the complex constraints in the air cargo loading scenario. However, combinatorial optimization methods are difficult to fully model complex constraints, resulting in low solution accuracy.

(2) *Analysis of different large models.* By observing the accuracy results of the five LLMs, we find that the accuracy of Deepseek is lower than that of other large models under different LLMs. This is because Deepseek lacks the knowledge of the relevant scenarios.

(3) *Heuristic search algorithm analysis.* We observe that without considering the algorithm runtime, the results of heuristic search achieve the highest solution accuracy in the evaluation of all aircraft types. This is because the heuristic search methods generate a huge search space to make the solution meet the complex constraints and improve the solution accuracy.

Comparison of algorithm runtime. By observing the algorithm runtime of evaluation under four aircraft types, we find the following two important observations: (1) compared with the heuristic algorithm, although FastLodaer loses some accuracy, it has a higher improvement in algorithm runtime. This is because LLM effectively reduces the search space, improves search efficiency, and reduces the time cost of the algorithm through its own understanding of complex constraints; (2) although the combinatorial optimization methods have the lowest algorithm runtime, they cannot handle complex constraints, resulting in the lowest solution accuracy.

Results. *FastLoader maintains stable solution accuracy in multiple LLMs and multiple aircraft types, with an average deviation of no more than 1%. Compared with combinatorial optimization methods, FastLoader has an average improvement of 10% in accuracy. Compared with the heuristic search methods, FastLoader has a loss of no more than 1.5% in accuracy, but the algorithm runtime is reduced by an average of 90%, which greatly reduces the time cost of calculating the solution.*

4.3. Evaluation of runtime optimization

Settings. In this comparison, we break down each step of the heuristic search and evaluate the algorithm runtime separately. We aim to analyze in depth which step of the heuristic search is optimized by Fastloader. Following the hyperparameter settings in the previous evaluation, we choose four heuristic search methods as baselines to conduct the experiment.

Comparison of runtime. Figures 3 and 4 show the algorithm runtime across four aircraft types. We have two key observations as follows: (1) the evolution phase accounts for 99% of the heuristic search runtime. The main reason why the heuristic algorithm consumes too much time is that the search speed in the evolution phase is slow, and it takes a long time to get an accurate solution; (2) by simplifying the search space with LLM, FastLoader significantly reduces the runtime of the algorithm in the evolution phase. This is because the proportion of feasible solutions in the search space increases, while the overall size of the search space decreases, so the runtime of solving the problem is effectively reduced.

Results. *By breaking down the heuristic search steps, we observe that FastLoader makes effective optimizations to the runtime during the evolution phase, reducing the runtime by 90%.*

4.4. Module ablation analysis

Settings. In this comparison, we set up ablation experiments for the three modules of FastLoader to evaluate the impact of each module on this method.

Iteration analysis. As shown in Figure 5a-c, we set different numbers of iterations in the evaluation, and we have two key observations as follows: (1) when FastLoader has a small number of iterations (e.g., 2 and 10 iterations), the balancing results %MAC and %index still achieve high accuracy; (2) we particularly observed that when the number of iterations is low, the solution accuracy of each large model is basically close. When the number of iterations exceeds 10, in the evaluation performance of %MAC, the results of Deepseek, phi, and llama are better than the Qwen, which shows these three LLMs have better capabilities in simplifying the load balancing search space under complex constraints.

Token length analysis. As shown in Figure 5 d-f, we set different length of token in the evaluation, and we have two key observations: (1) the accuracy of the solution drops significantly when the token is between 64 and 128 and tends to converge in other token settings (256, 512); (2) FastLoader has high accuracy and small differences under different token length settings. As a result, FastLoader is robust to token length settings. The reason is that all methods almost overlap when the token length is greater than 128 (the difference is less than 0.05% MAC). We conclude that whether it is Deepseek/phi based on instruction fine-tuning or the larger Qwen/Llama, as long as the length of the token exceeds 128, they stably output the optimal solution in the same order of magnitude.

Population size analysis. As shown in Figure 5 g-i, we set different population sizes in the evaluation, and we have the following three key observations: (1) FastLoader achieves high accuracy in the results of %MAC and %index under different initial population size settings; (2) when the population size exceeds 50, the solution accuracy of all LLMs further improves, but when the population size exceeds 100, the accuracy of Qwen3 decreases. This is because when Qwen3 generates the initial population, the similarity of the solutions in the population is too high, causing the result to fall into the local optimal solution. Under the same settings, other LLMs do not fall into the local optimal solution; (3) in terms of algorithm runtime, when the population size is set to 200, except for the time reduction of Qwen3, the algorithm runtime of the other LLMs increases. However, the overall time to complete

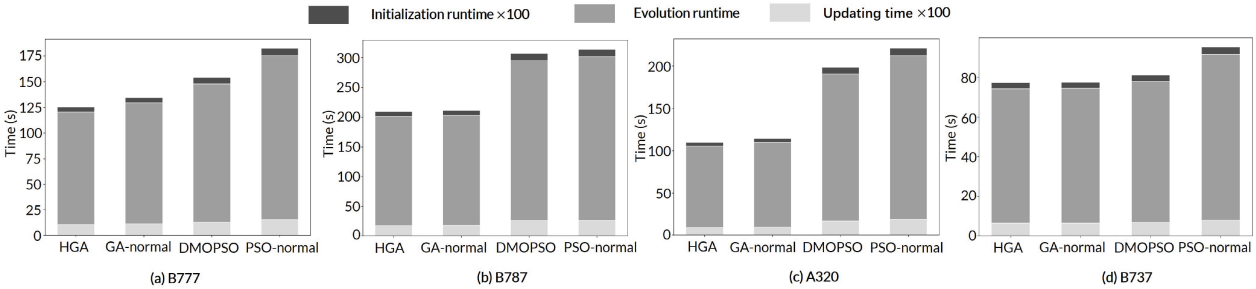

Figure 3. Experimental results of runtime evaluation on heuristic search.

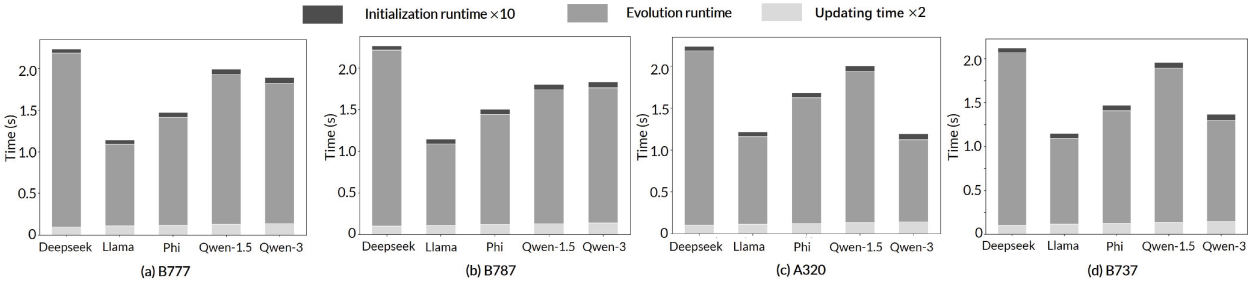

Figure 4. Experimental results of runtime evaluation on FastLoade.

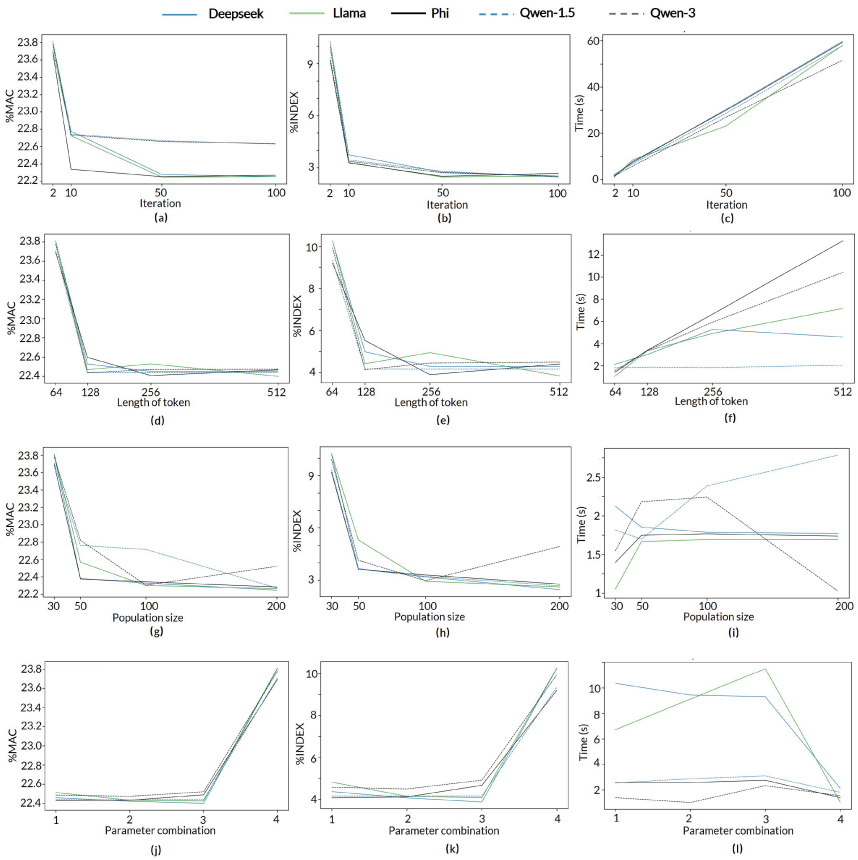

Figure 5. Experimental results of module ablation analysis.

the inference does not exceed 3 s, indicating that in different large model deployments, the optimization ability of FastLoader is robust under different population size settings.

Other parameter setting analysis. As shown in Figure 5j-l, we evaluate four different combinations of temperature, top_p and repeat penalty. The four parameter combinations are: {0.8,0.9,1.2}, {0.3,0.3,1.0}, {0.6,0.7,1.5}, and {0.5,0.6,1.1}. We have the following two key observations: (1) in terms of algorithm runtime, the runtime of Deepseek and Llama improves in the first three sets of parameter settings. However, for the other LLMs, FastLoader still has stable runtime under multiple experimental combinations and still has good robustness; (2) in the first three sets of parameter settings, the solution accuracy of LLM remains unchanged and has high robustness. But the accuracy decreases slightly under the fourth parameter combination. This is because under this parameter setting, temperature is 0.5, which introduces moderate generation diversity, while top_p is 0.6, which limits the selection space. They cause the model generation to deviate slightly from the optimal solution, but the accuracy remains at a high level.

Impact of Model Stability on Performance. We further analyze the impact of large model stability on our method's performance. In Stage one, modeling relies on prompts to help the model understand complex scenarios. Figure 5 d-f, j,k reflects this effect. When the length of tokens increases and the repetition penalty is too high, inference time rises and accuracy drops. This is because a high repetition penalty makes the model's judgment unstable in complex constrained settings. As a result, it fails to reduce the search space efficiently, lowering both speed and solution quality.

Results. *Across iterations, token lengths, population sizes, and other parameter settings, FastLoader maintains high solution accuracy within 3 s, confirming its robustness while exposing less than 1% differences among the evaluated LLMs.*

5. Conclusion

This work presents the design and evaluation of FastLoader, an LLM-based optimization approach for accelerating cargo loading. In the optimization problem modeling stage, cargo loading constructor accurately model the scenarios with numerous cargo types and complex constraints. It achieves higher solution accuracy than combinatorial optimization methods. In the search space reduction stage, the search space reducer effectively reduces the search time and improves the solution speed. It achieves a significantly shorter runtime while sacrificing only a negligible amount of solution accuracy.

We are further improving search efficiency in the solution process for cargo hold loading problems. This aims to handle more complex and diverse loading scenarios effectively. Complex scenarios often involve more constraints across multiple dimensions. Examples include item compatibility, stacking stability, and loading/unloading order. To obtain high-quality solutions in a shorter time, we aim to integrate reinforcement learning into the search space reduction process in the future.

Acknowledgments

None.

Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62272046, 62132019, 61872337), the high-quality development project of the Ministry of Industry and Information Technology (CEIEC-20240), Science and Technology Innovation Project of Beijing Institute of Technology (2023CX01017), and Open Project of Key Laboratory of the Ministry of Education (2023PY002).

Conflict of interest

The authors declare they have no competing interests.

Author contributions

Conceptualization: Zheng Chen, Chi Harold Liu, Rui Han
Investigation: Yunlai Cheng, Chi Harold Liu, Yinglong Wang
Methodology: Yunlai Cheng, Zhe Ouyang , Jing Chen
Formal analysis: Yunlai Cheng, Xinran Li , Yue Han
Writing – original draft: Yunlai Cheng, Zheng Chen, Rui Han
Writing – review & editing: Siqi Du, Ying Guo , Dongzhou Zhao, Meng Yu

Availability of data

Not applicable.

AI tools statement

All authors confirm that no AI tools were used in the preparation of this manuscript.

References

1. Karunathilake AN, Fernando A. Identifying the key influencing factors for the growth of air cargo demand. *J Glob Oper Strateg Sourc.* 2024;17(2):368-383.
2. Nath P, Upadhyay RK. Reformation and optimization of cargo handling operation at Indian air cargo terminals. *J Air Transp Res Soc.* 2024;2:100022.
3. Du Y, Chen F, Fan X, Zhang L, Liang H. Research on cargo-loading optimization based on genetic and fuzzy integration. *J Intell Fuzzy Syst.* 2021;40(4):8493-8500.
4. Brandt F, Nickel S. The air cargo load planning problem: a consolidated problem definition and literature review on related problems. *Eur J Oper Res.* 2019;275(2):399-410.
5. Mesquita AC, Sanches CA. Air cargo load and route planning in pickup and delivery operations. *Expert Syst Appl.* 2024;249:123711.
6. Zhao X, Dong Y, Zuo L. A combinatorial optimization approach for air cargo palletization and aircraft loading. *Mathematics.* 2023;11(13):2798.
7. Gajda M, Trivella A, Mansini R, Pisinger D. An optimization approach for a complex real-life container loading problem. *Omega (Oxford).* 2022;107:102559.
8. Dahmani N, Krichen S. On solving the bi-objective aircraft cargo loading problem. In: *Proc ICMSAO 2013.* IEEE; 2013:1-6.
9. Dahmani N, Krichen S. Solving a load balancing problem with a multi-objective particle swarm optimisation approach: application to aircraft cargo transportation. *Int J Oper Res.* 2016;27(1-2):62-84.
10. Chenguang Y, Hu L, Yuan G. Load planning of transport aircraft based on hybrid genetic algorithm. In: *Proc MATEC 2018.* Vol 179. EDP Sciences; 2018:01007.
11. Zhu L, Wu Y, Smith H, Luo J. Optimisation of containerised air cargo forwarding plans considering a hub consolidation process with cargo loading. *J Oper Res Soc.* 2023;74(3):777-796.
12. Bi X, Chen D, Chen G, et al. Deepseek LLM: scaling open-source language models with longtermism. *arXiv preprint* arXiv:2401.02954. Published 2024.
13. Achiam J, Adler S, Agarwal S, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774.* Published 2023.
14. Zhang S, Roller S, Goyal N, et al. OPT: open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068.* Published 2022.
15. Naveed H, Khan AU, Qiu S, et al. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435.* Published 2023.
16. Chang Y, Wang X, Wang J, et al. A survey on evaluation of large language models. *ACM Trans Intell Syst Technol.* 2024;15(3):1-45.
17. Huang S, Yang K, Qi S, Wang R. When large language model meets optimization. *Swarm Evol Comput.* 2024;90:101663.
18. Yang C, Wang X, Lu Y, et al. Large language models as optimizers. *arXiv preprint arXiv:2309.03409.* Published 2023.
19. Kaluzny BL, Shaw RD. Optimal aircraft load balancing. *Int Trans Oper Res.* 2009;16(6):767-787.
20. Macalintal JMV, Ubando AT. Optimal aircraft payload weight and balance using fuzzy linear programming model. *Chem Eng Trans.* 2023;103:613-618.
21. Zhao X, Yuan Y, Dong Y, Zhao R. Optimization approach to the aircraft weight and balance problem with the centre of gravity envelope constraints. *IET Intell Transp Syst.* 2021;15(10):1269-1286.
22. Mongeau M, Bes C. Optimization of aircraft container loading. *IEEE Trans Aerosp Electron Syst.* 2003;39(1):140-150.
23. Agbas E, Kusakci AO. A simulation approach for aircraft cargo loading considering weight and balance constraints. *Int J Bus Ecosyst Strateg.* 2021;3(1):21-31.
24. Vancroonenburg W, Verstichel J, Tavernier K, Vanden Berghe G. Automatic air cargo selection and weight balancing: a mixed integer programming approach. *Transp Res E Logist Transp Rev.* 2014;65:70-83.
25. Limbourg S, Schyns M, Laporte G. Automatic aircraft cargo load planning. *J Oper Res Soc.* 2012;63(9):1271-1283.
26. Traversa FL. Aircraft loading optimization: memcomputing the 5th Airbus problem. *arXiv preprint arXiv:1903.08189.* Published 2019.
27. Yan S, Shih YL, Shiao FY. Optimal cargo container loading plans under stochastic demands for air express carriers. *Transp Res E Logist Transp Rev.* 2008;44(3):555-575.
28. Lurkin V, Schyns M. The airline container loading problem with pickup and delivery. *Eur J Oper Res.* 2015;244(3):955-965.
29. Totamane R, Dasgupta A, Rao S. Air cargo demand modeling and prediction. *IEEE Syst J.* 2012;8(1):52-62.
30. Nayak V, Sahu V. Quantum approach to optimize aircraft cargo loading. In: *Proc Int Conf Trends Quantum Comput Emerg Bus Technol (TQCEBT).* IEEE; 2022:1-6.
31. Shunzhi Z, Wenxing H. An improved optimization algorithm for the container loading problem. In: *Proc WRI 2009.* Vol 2. IEEE; 2009:46-49.
32. Sahun Y, Sikirda Y, Tymochko O. Application of computer load optimization model in an aircraft load planning process. In: *Encyclopedia of Information Science and Technology*, Sixth Edition. IGI Global; 2025:1-20.
33. Heidelberg KR, Parnell GS, Ames JE IV. Automated air load planning. *Nav Res Logist.* 1998;45(8):751-768.

34. Yan S, Lo CT, Shih YL. Cargo container loading plan model and solution method for international air express carriers. *Transp Plan Technol.* 2006;29(6):445-470.
35. Wong EY, Mo DY, So S. Closed-loop digital twin system for air cargo load planning operations. *Int J Comput Integr Manuf.* 2021;34(7-8):801-813.
36. Hao Z, Du Q. Civil aircraft stowage based on genetic al gorithm. *China Sci Pap.* 2021;16(8).
37. Yu Q, Qu S, Peng Z, Ji Y. The robust maximum expert consensus model considering satisfaction preference. *J Ind Manag Optim.* 2025;21(3):2416-2455.
38. Zhu K, Qu S, Ji Y, Ma Y. Distributionally robust chance constrained maximum expert consensus model with incomplete information on uncertain cost. *Group Decis Negot.* 2024:1-41.
39. Guo D, Yang D, Zhang H, Song J, Zhang R, Xu R, Zhu Q, Ma S, Wang P, Bi X, et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*. Published 2025.
40. Touvron H, Martin L, Stone K, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*. Published 2023.
41. Abdin M, Aneja J, Awadalla H, et al. Phi-3 technical report: a highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*. Published 2024.
42. Bai J, Bai S, Chu Y, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*. Published 2023.
43. Yang A, Li A, Yang B, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*. Published 2025.
44. Tseremoglou I, Bombelli A, Santos BF. A combined forecasting and packing model for air cargo loading: a risk-averse framework. *Transp Res E Logist Transp Rev.* 2022;158:102579.

Yunlai Cheng *is a PhD student at Beijing Institute of Technology..*
https://orcid.org/0009-0005-3673-211X

Zheng Chen *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0000-9304-2047

Siqi Du *is a PhD student at Beijing Institute of Technology.*
https://orcid.org/0009-0000-9242-6241

Meng Yu *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0002-7231-8934

Dongzhou Zhao *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0000-8016-5603

Xinran Li *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0006-1206-1523

Yue Han *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0003-0794-7029

Zhe Ouyang *is a staff at TravelSky Technology Limited.*
https://orcid.org/0009-0008-8090-1920

Yinglong Wang *is a professor at Shandong Computer Science Center.*
https://orcid.org/0000-0002-8350-7186

Jing Chen *is an associate professor at Shandong Computer Science Center.*
https://orcid.org/0000-0001-5689-4742

Ying Guo *is a professor at Shandong Computer Science Center.*
https://orcid.org/0000-0002-4262-874X

Chi Harold Liu *is a professor at Beijing Institute of Technology.*
https://orcid.org/0000-0002-0252-329X

Rui Han *is an associate professor at Beijing Institute of Technology.*
https://orcid.org/0000-0001-6894-1921

An International Journal of Optimization and Control: Theories & Applications
(https://accscience.com/journal/ijocta)