Engineering xxx (xxxx) xxx

Contents lists available at ScienceDirect

Engineering

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

Research
Applications of ChatGPT—Article

Domain-Specific Large Language Model for Maintenance Decision-Making on Wind Farms by Labeled-Data-Supervised Fine-Tuning

Dongming Fan [a,†], Meng Liu [a,†], Yi Shao [b], Linchao Yang [b,*], Yiliu Liu [c], Yue Zhang [a], Yi Ren [a], Zili Wang [a]

[a] School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China
[b] School of Economics and Management, North China Electric Power University, Beijing 102206, China
[c] Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology, Trondheim 7491, Norway

ARTICLE INFO

Article history:
Received 31 May 2025
Revised 26 November 2025
Accepted 13 December 2025
Available online xxxx

Keywords:
Large language model
Maintenance decision-making
Wind farms
Fine-tuning

ABSTRACT

Wind farm operators always need a better maintenance strategy to increase resource utilization efficiency while controlling operation and maintenance costs. However, conventional maintenance decision-making approaches are time-consuming and have poor flexibility and adaptability to various scenarios. This study addressed these challenges by using a large language model (LLM) to understand, generate, and plan maintenance strategies for wind farms characterized by various failure modes and maintenance costs. A labelled-data-supervised fine-tuning LLM for maintenance, named LLM4M, is proposed. The proposed LLM4M model is trained on an extensive dataset of mathematical programs for maintenance to generate optimal strategies for wind farms. Compared with other large parameter LLMs, the fine-tuned LLM4M model demonstrates remarkable accuracy, with an error of approximately 2% from the optimal strategy. In addition, the generalization of the proposed LLM4M model has achieved remarkable results. If the LLM4M model correctly generates the maintenance strategy, the maintenance cost deviates from the optimal solution by only approximately 5%. Furthermore, phase transition behavior is observed, which provides considerable guidance for the development of domain-specific LLMs for the maintenance domain.

1. Introduction

In the 21st century, global society has entered a clean energy and low-carbon transition period [1]. Among the various renewable energy resources, wind power has the advantages of mature technology, relatively long use experience, and low cost [2]. By the end of 2022, the newly installed capacity of wind power reached 85.7 GW globally and 37.63 GW in China as a single market. According to the Global Wind Energy Association (GWEC) "Global Wind Power Report 2022" prediction data, the newly installed capacity of global wind power will continue to maintain a stable growth rate of approximately 6.6% per year in the future, and it is expected to approach 128.8 GW by 2026 [1].

Although wind power is environmentally benign, the operational availability of wind farms remains constrained. First, the inherent intermittency of wind resources fundamentally restricts continuous power generation, and second, wind turbines operating under severe environmental conditions are highly susceptible to multiple failure modes [3,4]. These technical and environmental challenges collectively reduce system reliability and operational uptime and, in turn, increase the requirements for frequent maintenance of wind turbines. Operation and maintenance (O&M) expenditure accounts for 30% of the total life cycle cost (LCC) of onshore wind farms [2,5]. Consequently, wind farm operators always need an optimal maintenance strategy to increase resource utilization efficiency while controlling O&M costs.

Two principal approaches exist for formulating wind farm maintenance strategies: conventional analytical [6–14] and machine learning (ML) techniques [15–22]. Considerable research efforts have been dedicated to developing optimized maintenance strategies through analytical approaches, including system modeling and simulation, mathematical programming, and heuristic algorithms. All these approaches have demonstrated their ability to generate maintenance schedules for wind energy systems. The system modeling and simulation approach analyses component interactions and failure propagation mechanisms. This

* Corresponding author.
E-mail address: yanglinchao@ncepu.edu.cn (L. Yang).
† These authors contributed equally to this work.

https://doi.org/10.1016/j.eng.2025.12.019

methodology emphasizes reliability assessment and lifecycle cost evaluation through virtual system representations [7–9]. Mathematical programming adopts rigorous optimization frameworks (e.g., mixed-integer linear programming (MILP) and dynamic programming) to achieve optimality under defined operational constraints, prioritizing precise resource allocation and temporal coordination of maintenance activities [10,11]. Heuristic algorithms, particularly metaheuristic techniques such as genetic algorithms and particle swarm optimization, provide computationally efficient solutions for high-dimensional optimization problems, sacrificing mathematical optimality for enhanced computational tractability and real-world adaptability [12–14]. Notably, these analytical approaches tend to be time-consuming and have poor flexibility in various scenarios.

Owing to advances in machine learning, particularly deep learning (DL), the field of maintenance decision-making for wind farms has been revolutionized and has witnessed a shift to data-driven approaches. The ML-based approach typically comprises two main types: remaining useful life (RUL) prediction and maintenance strategy decision-making. Methods based on artificial neural networks [17] and support vector machines [18] have been extensively utilized for specifying maintenance strategies, highlighting the possibility of intelligent maintenance. However, the performance of these ML-based models strongly depends on high-quality data, which are often laborious and difficult to acquire. Moreover, such approaches often struggle to handle big data, particularly when facing high-dimensional variables that exhibit strong intervariable and intra-variable nonlinearity.

Recently, DL models have achieved remarkable progress across various domains owing to the exceptional ability of automatic feature extraction and robust nonlinearity modeling [19]. This characteristic positions DL as an ideal candidate for wind farm maintenance decision-making, particularly in scenarios of long-term or sequential maintenance decision-making. By leveraging DL methods in maintenance decision-making, intricate system features and failure modes can be effectively extracted from the raw monitoring data, thereby facilitating smart maintenance. Numerous researchers have developed various DL-based methods, such as deep Q-network [20], convolutional neural network [21], recurrent neural network [21], random forest and transformer models [22], and have effectively applied them in maintenance decision-making. However, the performance of these DL-based models strongly depends on the trained model, which usually has fixed inputs and outputs. Moreover, these models are highly sensitive to the scenarios. Different scenarios require different DL models, meaning that DL-based models have insufficient adaptability and flexibility.

Recently, pretrained large language models (LLMs) have demonstrated excellent understanding and interpretability [23,24] and have become powerful tools for natural language understanding and task generation. Owing to their advanced comprehension and few-shot learning capability, pretrained LLMs can resolve many domain-specific problems, such as fault diagnosis [25,26], RUL prediction [27], risk assessment [28,29], and biochemical materials research [24,30,31]. However, to our knowledge, maintenance decision-making using pretrained LLMs is still lacking. This deficiency exists possibly because although LLMs exhibit strong generation capability in language-based tasks, their planning capability for maintenance planning tasks—which involve nonlinguistic status information, complex failure modes, and sequential decision processes—remains uncertain and underexplored.

The fine-tuning of LLMs within specific vertical domains has gained widespread acceptance across various professional domains. In the clinical field, LLMs are being fine-tuned to enhance medical diagnosis and treatment planning [32,33]. In the financial field, fine-tuning LLMs enables more accurate risk assessment and investment decision-making [34]. In the legal domain, fine-tuning LLMs helps in legal document analysis and case strategy formulation [35]. Domain-specific fine-tuning of pretrained LLMs enhances their capacity to comprehend specialized tasks and yields strong performance in vertical domain applications.

Considering the potential of LLMs in maintenance decision-making and the research gap, this study proposes an LLM-enabled solution for allocating maintenance teams and maintenance route planning on wind farms. The core of the method is an LLM trained on an extensive dataset of precise solution models for maintenance. The LLM model is used to capture the complex and potential relationships between the failure modes of turbines, the capability of the maintenance team, the number of maintenance teams, and the layout of wind farms. By leveraging LLM models, this study demonstrates the potential for maintenance decision-making on wind farms across a wide range of wind farm layouts and failure conditions, which is particularly valuable for wind farm operation and maintenance. The expected contributions of this study are as follows.

- First, pretrained LLMs and corresponding fine-tuning approaches in the domain of maintenance decision-making are introduced.
- A domain-specific LLM is created by fine-tuning a pretrained LLM based on labelled data, with a considerable improvement in the accuracy of the LLMs in maintenance decision-making within and without the fine-tuning data.
- Guidance on developing maintenance strategies using LLMs with observed phase transition behavior.

The remainder of this study is organized as follows: The procedure of the proposed LLM-enabled framework is detailed in Section 2. Section 3 proposes a prompt and fine-tuning approach based on low-rank adaptation (LoRA). Some experiments in fixed-scenario and nonfixed-scenario are proposed in Section 4. Conclusions and future research are presented in Section 5.

2. Problem descriptions and solution framework

2.1. Description of the wind turbine maintenance planning problem

Wind turbine maintenance scheduling can be characterized as a collaborative multiteam route optimization problem with three key features [36]. First, a wind farm can be served by multiple maintenance teams with heterogeneous service capabilities, where considerable differences exist in repair time and cost across teams for distinct failure modes (e.g., crack, damage, dirt and peeled paint). Second, wind turbines are spatially distributed, with geographic distances between devices determining travel costs, necessitating joint optimization of repair sequences and spatial transitions. Third, the problem focuses on single-objective cost minimization, aiming to achieve the lowest total cost (including operational and travel expenses) through optimized team routing and fault-handling strategies. Notably, this discussion exclusively pertains to corrective maintenance interventions conducted after equipment failure. This problem serves as an extension of the travelling salesman problem (TSP), which incorporates multiteam coordination, heterogeneous service constraints, and failure mode dependencies to form an optimization problem [37,38]. To effectively describe this problem, the symbols are defined as shown in Table 1.

Then, the optimization objective can be described as follows:

$$\min \sum_{k\in K}\left[\sum_{f\in V}\sum_{j\in V} c_k^{\mathrm{tra}} d_{fj} x_{fj}^k + \sum_{f\in F}\sum_{m\in M} c_{k,m}\delta_{fm} y_f^k\right] \tag{1}$$

Table 1
Definition of symbols.

Symbols	Meaning
K	Set of maintenance teams, $\|K\|$ is quantity of maintenance teams
F	Set of wind turbines requiring maintenance
M	Set of failure modes
$V = \{0\} \cup F$	All nodes V (including the set of nodes requiring maintenance F and the starting node 0)
d_{ij}	Distance travel from node i to node j
c_k^{tra}	Travel cost per unit distance for maintenance team k
v_k	Travel speed of maintenance team k
$c_{k,m}$	Repair cost for maintenance teams k to repair failure mode m
$t_{k,m}$	Repair time for crew k for failure mode m
$\delta_{fm} \in \{0,1\}$	If the failure mode of the wind turbine f is m, it is 1. Otherwise, it is 0.
$x_{fj}^k \in \{0,1\}$	If the team k travels directly from node f to j, it is 1. Otherwise, it is 0.
$y_f^k \in \{0,1\}$	If team k is responsible for maintaining wind turbine f, it is 1. Otherwise, it is 0

It aims to minimize the total travel and maintenance costs while ensuring that all wind turbines awaiting repair are accurately and completely assigned maintenance teams for inspection and repair. Specifically, the objective function comprises two components: the sum of travel costs incurred by each maintenance team based on travel distance (or time), and the sum of labor repair costs incurred by each team for different failure modes.

In maintenance scheduling, the following constraints exist:

- Maintenance task assignment constraint: Each wind turbine requiring maintenance must be assigned exclusively to one maintenance team, ensuring that no device is left unassigned and that no redundant assignments occur.

$$\sum_{k\in K} y_f^k = 1, \forall f \in F \tag{2}$$

- Visit consistency constraint: If a maintenance team is assigned to repair a specific device, it must visit that device node, and conversely, any visit to a node implies that the team is assigned to that wind turbine.

$$\sum_{j\in V} x_{fj}^k = y_i^k, \forall f \in F,\ k \in K \tag{3}$$

- Route connectivity constraint: Each maintenance team must begin and end its route at the central depot, forming a closed-loop path that traverses all assigned wind turbine nodes.

$$\sum_{j\in I} x_{0j}^k = 1, \forall k \in K \tag{4}$$

- Flow conservation constraint: At any intermediate node (excluding the depot), the number of times a team enters the node must equal the number of times it exits, thereby ensuring continuity of the route.

$$\sum_{i\in V} x_{if}^k = \sum_{j\in V} x_{fj}^k, \quad \forall f \in F, k \in K \tag{5}$$

- Non-exemptive assignment constraint: Each maintenance team must be assigned at least one repair task, preventing any crew from remaining idle and ensuring a balanced utilization of all available teams in the schedule.

$$\sum_{f\in F} y_f^k \geq 1, \quad \forall k \in K \tag{6}$$

- Subtour elimination constraint: Through the incorporation of sequence-indexed variables and the application of Miller–Tucker–Zemlin (MTZ) inequalities, the formation of disconnected subcycles among visited nodes is prevented, ensuring that each route constitutes a complete inspection tour. An additional variable $u_i^k, u_0^k = 0$ is introduced to set this constraint as follows.

$$u_i^k - u_j^k + |F| x_{ij}^k \leq |F| - 1, \forall i,j \in F, i \neq j, k \in K \tag{7}$$

$$1 \leq u_i^k \leq |F|, \quad \forall i \in F, k \in K \tag{8}$$

The aforementioned equations collectively form a mixed-integer linear programming model that integrates multivehicle routing planning with heterogeneous maintenance costs. This model not only reflects the disparities in team capabilities and fault types encountered in actual operations but also employs mathematical optimization techniques to derive cost-optimal scheduling solutions.

2.2. LLM-enabled solution framework for maintenance scheduling

LLMs have demonstrated exceptional generalization capabilities in solving complex problems, particularly in handling unstructured data, multimodal fusion, and dynamic decision-making scenarios [39]. Considering the limitations of traditional operational research methods in modeling flexibility and computational efficiency when practical challenges such as heterogeneous team coordination, spatiotemporal coupling constraints, and multi-failure mode interactions are addressed, this paper proposes a novel LLM-driven maintenance planning framework as shown in Fig. 1.

The proposed solution framework comprises the following standardized steps:

Step 1: Demand clarification. This step involves systematically mapping the spatial distribution of wind turbines and compiling a comprehensive list of units requiring maintenance to establish task scope and scale.

Step 2: Information extraction. Specific failure types, geographic coordinates, and technical capability parameters of available maintenance teams are retrieved from operational databases.

Step 3: Model interaction. Problem descriptions are provided to our fine-tuned LLM, named LLM4M, using structured prompt templates, enabling natural language interaction to ensure full model comprehension of maintenance objectives and constraints.

Step 4: Model reasoning. The LLM4M comprehends problem-specific contextual information and employs deductive reasoning to dynamically adjust task allocations under operational constraints, ultimately achieving measurable cost reductions through optimized resource scheduling.

Step 5: Solution delivery. Complete solutions are generated encompassing scheduling sequences, route optimization, and cost estimation, resulting in executable operational plans.

The cornerstone is to develop a domain-specific LLM tailored for maintenance planning tasks, complemented by a purpose-built prompt engineering framework. This process necessitates the construction of an extensive operational dataset to fine-tune the foundational model, enhancing its adaptability and decision-making capabilities within equipment maintenance contexts. Detailed formulations of the context-specific prompt templates and fine-tuning methods will be systematically elaborated upon in subsequent sections.

3. Methodology

In this study, the location information of the nodes requiring maintenance of different failure modes in the wind farm, fault information, target and the quotes and moving speeds of each maintenance team were filled into the prompt templates that had been designed in advance. This batch of data was subsequently

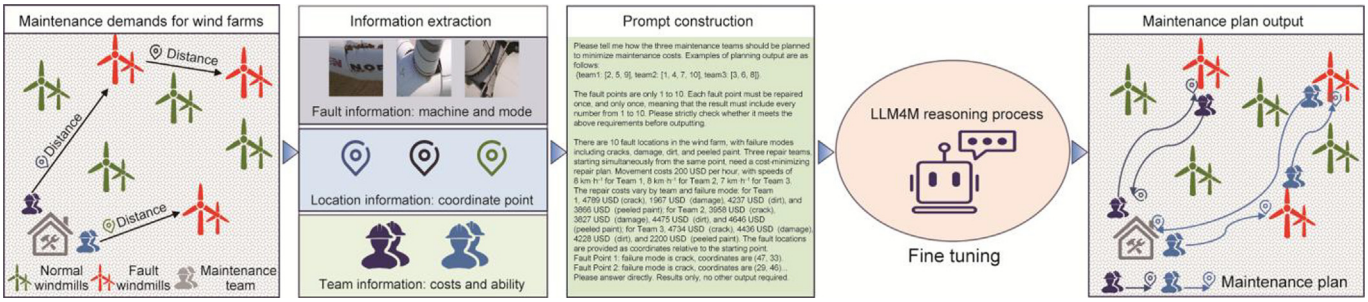

Fig. 1. Large language model-enabled maintenance planning solution framework.

employed to supervise and fine-tune certain parameters of the base model. Through continuous iteration, a dedicated LLM for wind power maintenance can be obtained.

3.1. Construction of prompt

The design of prompt is crucial in the application of LLM maintenance planning tasks [40,41]. It is mainly used to guide the LLM to generate constraints and output requirements that meet the corresponding maintenance problems. The prompt in this study mainly comprise four parts: problem scenario description, problem parameter filling, output example, and output constraint requirements. The prompt template is shown in Fig. 2.

When the problem scenario is described, namely, encoding the maintenance issue into the LLM input, the context of the problem, the faulty equipment involved, the basic situation of the maintenance team, and the final goal of the maintenance must be clearly and comprehensively presented. This approach ensures that the LLM can accurately understand the essence and core points of the problem.

When problem parameters are provided, a structured prompt template is designed, including multiple fillable positions. Relevant information about the maintenance team and the specific description of the faulty equipment are extracted and used to fill these positions in sequence according to the established order. This design enables the prompt template to be flexibly adjusted according to different scenarios. The introduction and examples of the content that needs to be filled in each slot are shown in Table 2.

In the context of output examples, the prompts of a few samples are designed to clearly specify the desired output format, thereby enabling the LLM to accurately present key information. Following prompt guidance, the output results from the LLM can be more readily recognized and efficiently extracted by the programmed code, facilitating subsequent automated processing. Additionally, the output examples incorporate certain constraints, which assist the LLM in understanding the core aspects and boundary requirements of the problem. To efficiently utilize the LLM and minimize redundant output contents, the prompts are designed to emphasize the direct output of the final result. This strategy not only considerably enhances output efficiency but also allows for rapid access to useful information.

The output constraints are the target constraints of the maintenance problem to be solved. In the tests of maintenance planning tasks, the output results of general LLMs are prone to missed repairs, repeated repairs, and so forth. To avoid this situation, the importance of these constraints must be emphasized in prompt to ensure that the output of the LLM meets the constraints of the problem. Fig. 3 shows an example of a prompt for a maintenance question generated using the prompt template.

3.2. Fine-tuning based on low-rank adaptation

This study primarily employs the LoRA method for supervised fine-tuning to adjust the LLM [42]. The core of this approach lies in freezing the original weight parameters of the pretrained model and achieving efficient parameter updates by introducing trainable low-rank decomposition matrices. A schematic of the LLM before and after fine-tuning is shown in Fig. 4.

Specifically, for the pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$,where $\mathbb{R}$ represents the set of real numbers. The parameter update form defined by LoRA is expressed as follows:

Please tell me how the [X_1] maintenance teams should be planned to minimize maintenance costs. Examples of planning output are as follows:
[X_2] **Output example**

The fault points are only 1 to [X_3]. Each fault point must be repaired once, and only once, meaning that the result must include every number from 1 to [X_3]. Please strictly check whether it meets the above requirements before outputting.

There are [X_3] fault locations in the wind farm, with failure modes including cracks, damage, dirt, and peeled paint. [X_1] repair teams, starting simultaneously from the same point, need a cost-minimizing repair plan.

Movement costs [X_4] per hour, [X_5], The repair costs vary by team and failure mode:
[X_6]

The fault locations are provided as coordinates relative to the starting point.
[X_7]

Problem scenario description

Please answer directly. Results only, no other output required. **Output constraint requirements**

Fig. 2. Prompt template for maintenance tasks.

Table 2
Slot definitions and examples used in maintenance task prompts.

Slot	Definition	Example
$[X_1]$	Number of maintenance teams	"3"
$[X_2]$	Output example	"{team1: [2,5,9], team2: [1,4,7,10], team3: [3,6,8]}"
$[X_3]$	Number of nodes requiring maintenance	"10"
$[X_4]$	Cost of moving the maintenance team	"200 USD"
$[X_5]$	The speed at which each maintenance team moves	"with speeds of 8 km·h^{-1} for Team 1, 8 km·h^{-1} for Team 2, 7 km·h^{-1} for Team 3"
$[X_6]$	Maintenance for each maintenance team for different failure modes	"for Team 1, 4789 USD (crack), 1967 USD (damage), 4237 USD (dirt), and 3866 USD (peeled paint); for Team 2..."
$[X_7]$	The location of each fault site	"Fault Point 1: Failure mode is crack, coordinates are (47, 33); Fault Point 2: failure mode is crack, coordinates are (29, 46); ..."

$$W = W_0 + \Delta W = W_0 + BA \tag{9}$$

In the above formula (9), $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ are trainable low-rank matrices, with the rank r satisfying the constraint $r \ll \min(d, k)$. This design considerably reduces the number of trainable parameters from the original $d \times k$ to $d \times r + r \times k$ while effectively preserving the knowledge integrity of the pretrained model.

During LoRA fine-tuning, the introduction of low-rank matrices substantially decreases the gradient computation and optimizer state storage requirements. Moreover, the strategy of freezing the original parameters effectively prevents catastrophic forgetting of pretrained knowledge. At deployment, the parameters can be merged through a simple matrix addition operation $W_0 + BA$, ensuring that no additional inference latency is introduced.

In the fine-tuning process, LoRA was applied to the attention mechanism-related layers of the model, including the query, key, value, and output projection layers, which are crucial for capturing complex dependencies between input sequences. Fig. 4 shows schematic of the process before and after LoRA fine-tuning in this study. The hyperparameters were set as follows: an LoRA rank of 8, a scaling factor (alpha) of 16, a dropout rate of 0.1 (to prevent overfitting), an optimizer set to Adam, a learning rate of 2×10^{-5}, a batch size of 4, and 2 training epochs. LoRA hyperparameter analysis was included in Appendix A Table S1. Additionally, to monitor the training process, the loss value was recorded every 10 steps, and after training, the loss curve was plotted to analyze model convergence.

4. Experiments and results

4.1. Details of the experiment

The experimental equipment and environment is configured as follows: The server uses an Intel Xeon Skylake architecture with a 20-core 40-thread processor, which is equipped with 144 GB of memory and 2 NVIDIA A100-PCIE-40GB graphics processing units (GPUs). The system architecture is x86_64, which supports 32-bit and 64-bit operation modes. The operating system is Linux. The software stack uses Python 3.12.9 as the programming language, and PyTorch version 2.6.0 is installed. The CUDA version is 12.4, which is compatible with the GPU.

The experiment involves fine-tuning small-parameter LLMs and conducting multiple rounds of testing by calling the application programming interfaces (APIs) of large-parameter LLMs available on the market. The small-parameter LLM selected for this purpose is Qwen2.5-7B-Instruct [43], which is open-sourced by the Alibaba Qianwen team. This model is considered one of the most accessible and easily fine-tunable models within a given parameter range, and it demonstrates a strong ability to understand various instructions [44]. Four of the most globally influential large parameter LLMs, ChatGPT, Claude, Qwen, and DeepSeek, are chosen for comparison because of their ready availability and ease of use. Information on these LLMs is shown in Table 3. The parameter values of the GPT-4o and Claude 3.5 Sonnet models are given by the Microsoft research team [45].

In this study, the cost deviation rate $R_{\text{deviation}}$, mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE),

Please tell me how the 3 maintenance teams should be planned to minimize maintenance costs.
Examples of planning output are as follows:
{team1: [2, 5, 9], team2: [1, 4, 7, 10], team3: [3, 6, 8]}

The fault points are only 1 to 10. Each fault point must be repaired once, and only once, meaning that the result must include every number from 1 to 10. Please strictly check whether it meets the above requirements before outputting.

There are 10 fault locations in the wind farm, with failure modes including cracks, damage, dirt, and peeled paint. 2–3 repair teams, starting simultaneously from the same point, need a cost-minimizing repair plan.

Movement costs 200 USD per hour, with speeds of 8 km·h^{-1} for Team 1, 8 km·h^{-1} for Team 2, 7 km·h^{-1} for Team 3. The repair costs vary by team and failure mode: for Team 1, 4789 USD (crack), 1967 USD (damage), 4237 USD (dirt), and 3866 USD (peeled paint); for Team 2...The fault locations are provided as coordinates relative to the starting point. Fault Point 1: Failure mode is crack, coordinates are (47, 33). Fault Point 2: Failure mode is crack, coordinates are (29, 46)...

Please answer directly. Results only, no other output required.

Fig. 3. Example of a fault task prompt generated using the prompt template.

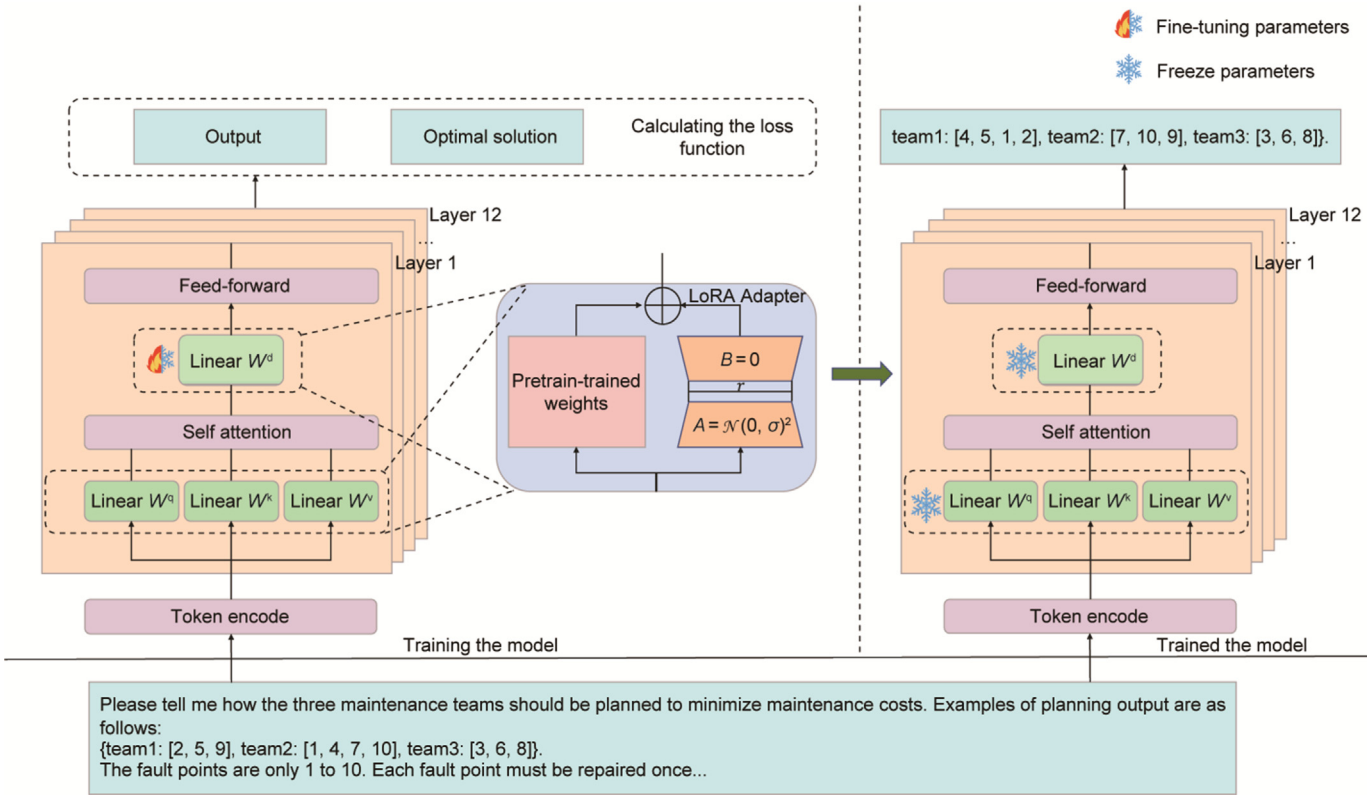

Fig. 4. Schematic of the process before and after LoRA fine-tuning in this study. Matrix A serves as the dimensionality reduction projection matrix and is initialized according to a normal distribution with a mean of 0 and a standard deviation of σ, where $\mathcal{N}$ represents the normal distribution. Matrix B serves as the dimensionality expansion projection matrix and is initialized as a zero matrix. W^q, W^k and W^v, are the weight matrices used in the Transformer's self-attention mechanism to map the input to the Query, Key, and Value vectors, respectively. W^d, is the down-projection weight matrix in the Feed-Forward Network of the Transformer. The user inputs the prompt template into the LLM for model fine-tuning. The model learns the solution for the specified maintenance task on the basis of the label and finally outputs the result directly.

Table 3
Information about the large-parameter LLM used for comparative experiments.

Model Name	Model version	Vendor	Parameter scale	Usage methodology
GPT	GPT-4o	OpenAI	200B (estimate)	API
Claude	Claude 3.5 Sonnet	Anthropic	175B (estimate)	API
DeepSeek	DeepSeek-V3	Deepseek-AI	671B	API
Qwen	Qwen2.5-72B	Qwen	72B	API

and mean absolute percentage error (MAPE) are selected as the main evaluation indicators to comprehensively measure the prediction accuracy and cost-effectiveness of the model.

The cost deviation is used to measure the difference between the actual cost and the optimal cost. The calculation formula is as follows:

$$R_{\text{deviation}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{C_i - O_i}{O_i}\right) \times 100\% \tag{10}$$

The MSE is used to measure the difference between the model output and the true value. The calculation formula is as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(C_i - O_i)^2 \tag{11}$$

The RMSE is used to measure the difference between the model output and the true value. It is the square root of the MSE. Compared with the MSE, the unit of the RMSE is consistent with the unit of the original data and is easier to interpret. The calculation formula is as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(C_i - O_i)^2} \tag{12}$$

The MAE is used to measure the mean absolute difference between the model output and the true value. The MAEs have the same weight for all the errors and are unaffected by extreme values. The calculation formula is as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|C_i - O_i| \tag{13}$$

The MAPE is used to measure the average value of the absolute difference between the model output and the true value. The MAPE reflects the proportion of error in the true value and is suitable for comparing data of different dimensions. The calculation formula is as follows:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{O_i - C_i}{C_i}\right| \tag{14}$$

In the above formulas (10)–(14), n is the total number of samples; C_i represents the cost of the generation strategy for the i-th sample, and O_i represents the cost of the optimal strategy for the i-th sample.

4.2. Performance results in fixed-scenario maintenance tasks

4.2.1. Fixed scenario description and dataset

To evaluate whether fine-tuning can enhance the performance of LLMs in maintenance planning tasks and to determine whether the LLMs have successfully learned the solution ideas and logic of the maintenance problem presented in this paper, we designed a series of comparative experiments.

For the fine-tuning dataset, an integer programming model is constructed on the basis of the problem description in Section 2 to create a fixed scenario. In this scenario, there are three maintenance teams, and each maintenance task involves five nodes that need to be repaired. Each maintenance team starts from the origin, follows a specific route, and eventually returns to the origin. All maintenance teams are required to perform the maintenance tasks. The stochastically generated datasets were designed to simulate a broader spectrum of maintenance scenarios under fixed team and failure conditions, thereby rigorously testing the adaptability of the model to dynamic operational challenges. The range of values of maintenance-related parameters for fixed scenarios in the dataset is shown in Table 4 below, where references [46–48] provide the basis for generating experimental data. The location, fault type, and maintenance cost of the windmill were randomly generated, and the problem was accurately solved to construct a dataset of fixed-scenario maintenance tasks. The dataset comprises 10 000 samples in the training set and 1 000 samples in the test set.

We compare the performance of the fine-tuned model, the non-fine-tuned model, and the large-parameter model on the fixed maintenance planning problem involving three maintenance teams and five fault sites. This comparison allows us to assess the impact of fine-tuning on model performance and whether the fine-tuned model can better understand and solve the maintenance planning problem.

4.2.2. Performance of the LLM4M in the fixed-scenario

The Qwen2.5-7B-Instruct model is first fine-tuned on the training dataset and evaluated on the test dataset. For comparative analysis, the non-fine-tuned Qwen2.5-7B-Instruct model was also evaluated on the same test dataset to serve as a baseline. The performance of the model is evaluated in two dimensions:

Generation capability—assessing whether the model exhibits hallucinations and the validity of the maintenance plans it generates.

Planning capability—evaluating whether the cost of the proposed plan is the lowest or approaches to the lowest cost.

Fig. 5 illustrates the difference in generation capability between the non-fine-tuned and fine-tuned models under the fixed scenario. In 1000 test cases, the fine-tuned model consistently generates implementable maintenance plans, whereas the non-fine-tuned model does so with a probability of 83.40%. An implementable plan refers to a feasible solution that satisfies all problem constraints and can be executed in practice; it may not be optimal but remains valid and actionable. Nonimplementable plans are systematically categorized into two types. The first type, "incorrect output," accounts for 13.60% of the test cases and indicates that the model experienced hallucinations and failed to comprehend the maintenance planning task within the fixed scenario as instructed on the provided prompt. The second type, "invalid plan," occurs in 3.00% of the test cases, reflecting that although the model understood the task, the generated plans were nonimplementable, resulting in problems such as redundant or missing repairs.

Table 4
Parameters related to maintenance for the fixed scenario.

Parameters	Value	Unit
Number of maintenance team	3	—
Number of turbines required maintenance	5	—
Number of failure modes	4	—
Travel costs	200	USD·h^{-1}
Maintenance cost on turbines for various failure modes	[2000, 5000]	USD
Time-consuming on maintenance for various failure modes	[60, 180]	min
Speed for maintenance teams	[25,30,31]	km·h^{-1}

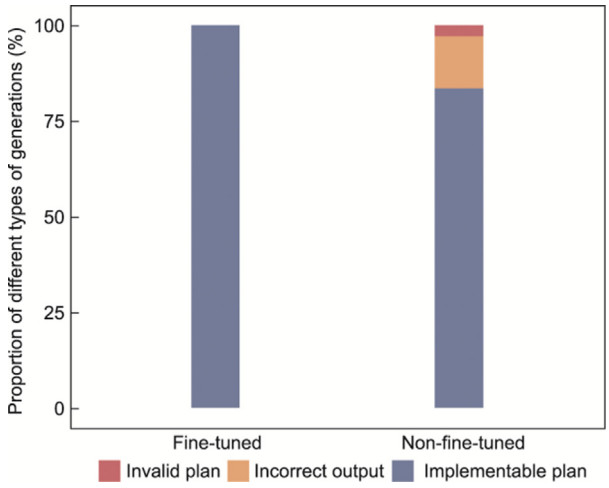

Fig. 5. Comparison of maintenance plan generation performance between the fine-tuned and non-fine-tuned models under the fixed scenario.

The total cost of each implementable plan generated by the model is calculated on the basis of the conditions provided in the prompt and compared with the cost of the optimal solution. Fig. 6 compares the cost deviations of all implementable plans generated by the fine-tuned and non-fine-tuned models. As shown, the results from the fine-tuned model clearly exhibit a thin-tailed distribution, with a lower median and standard deviation. The median and standard deviation of the cost deviations from the optimal solutions are 0.88% and 2.90%, respectively. These results demonstrate that under the fixed scenario, the fine-tuned model has a strong ability to generate maintenance plans that closely approximate the optimal cost. In contrast, the cost deviations of implementable plans generated by the non-fine-tuned model reach nearly 60% and show a significantly more dispersed distribution. Table 5 compares the cost deviations by MSE, RMSE, MAE, and MAPE between implementable plans generated by the fine-tuned and non-fine-tuned models and the optimal plan.

To further assess the effectiveness of the fine-tuned model, several widely recognized large-parameter LLMs with superior performance were evaluated on the same test dataset. As shown in Table 6, the proportion of implementable plans generated by LLM4M and the large-parameter LLMs reached or approached 100% in the test cases. This finding indicates that under the fixed scenario, the LLM4M and the large-scale LLMs can generate implementable maintenance plans. Furthermore, the performance of LLM4M is comparable to that of these LLMs and even exceeds that of Claude and Qwen.

In terms of planning capability, the LLM4M outperformed the large-parameter LLMs. As shown in Fig. 7, under the fixed scenario,

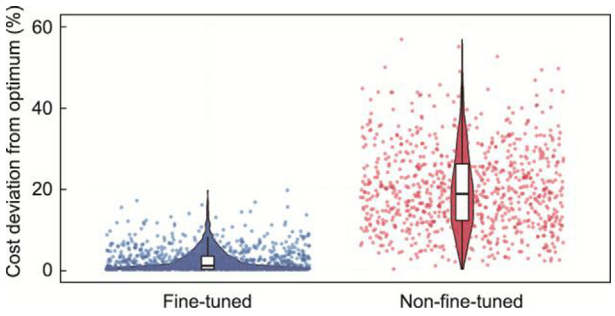

Fig. 6. Comparison of cost deviations within implementable plans generated by the fined-tuned and non-fine-tuned models under the fixed scenario.

the cost deviations between the implementable plans generated by LLM4M and the optimal plan were much lower than those of the four large-parameter LLMs. LLM4M had a lower median and standard deviation and a more concentrated distribution. In contrast, compared with LLM4M, the four LLMs exhibited more dispersed distributions, with maximum deviations exceeding 60% and much higher median medians and standard deviations. As shown in Fig. 8 and Table 7, compared with the large-parameter LLMs, the LLM4M achieves much lower values of MSE, RMSE, MAE, and MAPE. These results have practical importance. LLMs are already being applied or are gradually being adopted in a wide range of industries. For maintenance planning tasks, users without a background in mathematical modeling often seek more convenient ways to obtain implementable planning, and LLMs offer a promising method. However, the maintenance plans generated by commonly accessible LLMs (such as the four LLMs used in this experiment) fall far short of being optimal. As indicated by the MAE and MAPE values, users may incur up to an additional 4000 USD in cost. In contrast, the plans generated by LLM4M exhibit substantially lower RMSE, MAE, and MAPE values, indicating that it enables users to access maintenance plans with reduced additional cost and improved usability.

To further demonstrate the practicality of the LLM4M, we compared it with intelligent optimization models, including the genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and simulated annealing (SA). A task with three repair crews and five turbines required maintenance (T3F5) test set was used as the benchmark for this comparison. As shown in Fig. 9, LLM4M and ACO exhibit lower median and standard deviation values, indicating that their resulting distributions are more concentrated. In contrast, the distributions of the four other LLMs are more dispersed, with the maximum deviation exceeding 35%. Furthermore, as illustrated in Fig. 10 and Table 8, the MSE, RMSE, MAE, and MAPE values for LLM4M and ACO are significantly lower than those of the large-parameter LLMs. These results have important practical implications. They demonstrate that the performance of LLM4M is comparable to that of the current state-of-the-art traditional optimization models, and it generates higher-quality solutions than most other algorithms do. In practical engineering applications, implementing an intelligent optimization model requires maintenance personnel to have a solid grasp of its key principles. In comparison, LLM4M can produce a similarly effective solution through a simple text prompt, making it a much more convenient way to obtain a usable maintenance plan.

4.3. Performance results in nonfixed-scenario maintenance tasks

4.3.1. Nonfixed scenario description and dataset

The experimental results show that the fine-tuned LLM performs well in handling maintenance planning tasks in fixed scenarios. To further evaluate its capabilities, experiments with nonfixed scenarios are designed to evaluate the generalization ability of the model in different scenarios.

For datasets with nonfixed scenarios, a synthetic dataset was created on the basis of the integer programming model constructed according to the problem description in Section 2. The number of maintenance teams and the number of nodes requiring repair in the synthetic dataset vary, which is distinct from the fixed scenario in the experiment involving three maintenance teams and five nodes. Additionally, the location, fault type, and maintenance cost of the windmills in the synthetic dataset are randomly generated, and the problem can be solved accurately.

Each maintenance team must start from the origin, perform maintenance tasks sequentially according to the planned schedule, and then return to the origin. All teams must participate in maintenance tasks. Furthermore, a mixed dataset (TRFR) is constructed, combining various configurations of maintenance teams and nodes. The characteristics of all the datasets are presented in Table 9. The relevant parameters of all the samples meet the value range specified in Table 4.

During fine-tuning, the use of LoRA technology in the experiment was continued, and the same hyperparameter settings were maintained. Following fine-tuning, the model was applied to each test set in sequence, and the performance of the model on each test set was recorded. Special attention was also given to the model's performance on the mixed dataset (TRFR) to assess its adaptability to diverse data.

4.3.2. Generalization performance of LLM4M

Real-world maintenance operations encounter highly complex and dynamic scenarios characterized by variable weather conditions, diverse turbine types, and other operational uncertainties. At the model formulation level, these factors manifest as parameter variations in maintenance optimization frameworks. For instance, environmental conditions directly influence maintenance team mobility rates, whereas turbine type disparities necessitate distinct failure mode models and crew capability adaptations. The number of concurrent maintenance tasks and available teams also fluctuates significantly across scheduling instances. To validate the robustness and generalizability of the LLM4M across these multifaceted conditions and nonfixed scenarios, we systematically generated extensive datasets featuring randomized parameters, including turbine failure types, spatial configurations, cost structures, team mobility constraints, and repair capabilities. These datasets were created under varying team availability and task quantity scenarios to ensure thorough coverage of operational realities. All fully trained models were evaluated on the TRFR dataset to assess their performance under nonfixed scenarios.

Table 5
Performance of the fine-tuned and non-fine-tuned models under the fixed scenario (based on the MSE, RMSE, MAE, and MAPE between the generated implementable plans and the optimal plan costs).

Model	MSE	RMSE	MAE	MAPE
Fine-tuned	**574 629.5433**	**758.0432**	**453.1927**	**0.0200**
Non-fine-tuned	20 579 482.5829	4 536.4615	4 107.8879	0.1580

Table 6
Proportion of different generation types of LLM4M and large-parameter LLMs under the fixed scenario.

Model	Implementable plan	Incorrect output	Invalid plan
LLM4M	100%	0	0
Claude	99.60%	0	0.40%
GPT	100%	0	0
DeepSeek	100%	0	0
Qwen	98.80%	0	1.20%

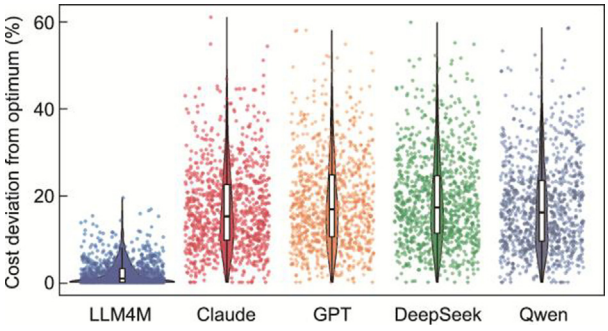

Fig. 7. Comparison of cost deviations within implementable plans generated by LLM4M and the LLMs under the fixed scenario.

Fig. 11 and Table 10 report the generation performance of LLM4M in nonfixed scenarios when trained on different datasets. In terms of generating implementable plans, the model trained on the T3F10 dataset outperforms the models trained on other datasets. LLM4M appears to be capable of understanding this type of maintenance planning problem through simple scenario comprehension. Models trained on datasets such as T3F10, T3F5, T3F9, T4F11, T2F5, T3F11, T6F13, and T5F10 generally exhibit stronger abilities in generating implementable plans than others do. In contrast, the model has a limited ability to understand more complex scenarios, as evidenced by its poor performance and severe hallucinations when trained on datasets such as T2F11, T2F12, and T4F17.

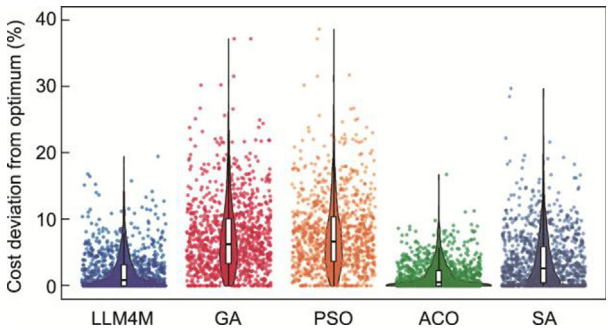

Fig. 9. Comparison of cost deviations within implementable plans generated by LLM4M and the traditional optimization model.

In terms of planning capability, Fig. 12 shows that models trained on datasets such as T5F10, T5F16, and T6F13 exhibit the most concentrated distribution of cost deviations between the generated executable plans and the optimal solutions when evaluated on the TRFR dataset. These models also achieve lower median devi-

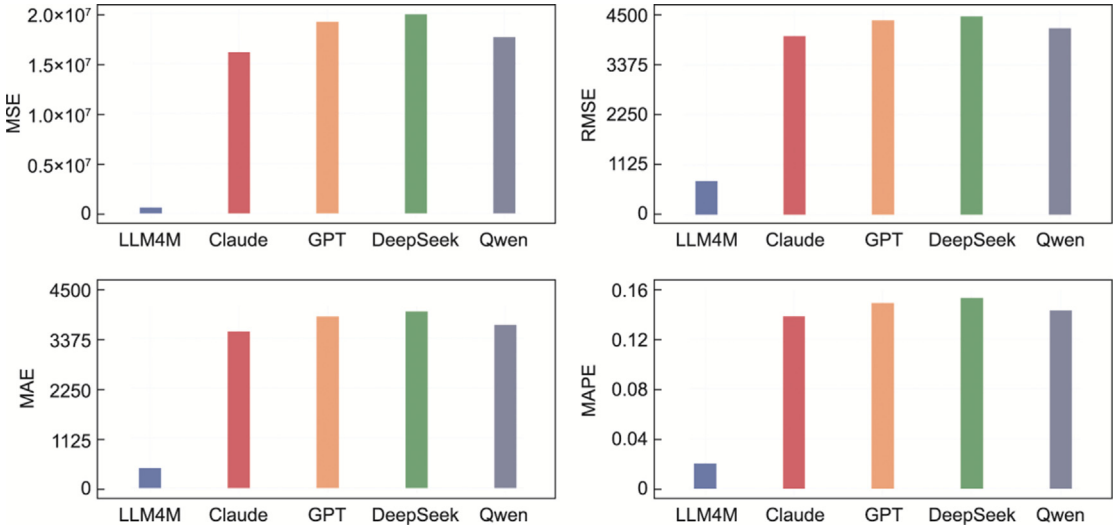

Fig. 8. Comparison of the performance of the LLM4M and the LLMs under the fixed scenario (based on the MSE, RMSE, MAE, and MAPE between the generated implementable plans and the optimal plan costs).

Table 7
Performance of the LLM4M and the LLMs under the fixed scenario (based on the MSE, RMSE, MAE, and MAPE between the generated implementable plans and the optimal plan costs).

Model	MSE	RMSE	MAE	MAPE
LLM4M	574 629.5433	758.0432	453.1927	0.0200
Claude	16 110 879.3859	4 013.8360	3 522.7647	0.1378
DeepSeek	19 907 225.5059	4 461.7514	3 971.0095	0.1526
GPT	19 140 917.7422	4 375.0335	3 855.6993	0.1485
Qwen	17 605 401.7107	4 195.8791	3 675.1583	0.1424

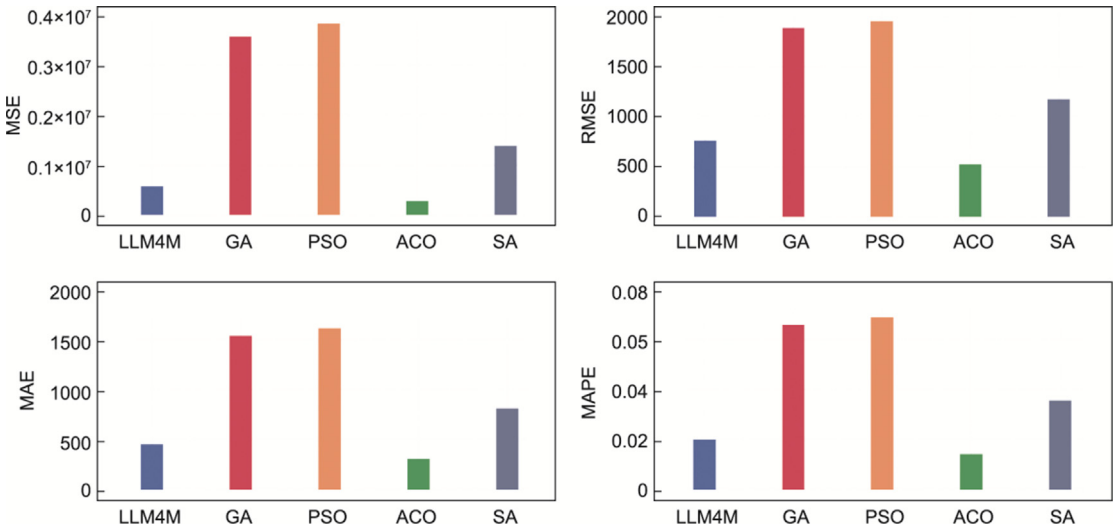

Fig. 10. Comparison of the performance of the LLM4M and traditional optimization models (based on the MSE, RMSE, MAE, and MAPE between the generated implementable plans and the optimal plan costs).

Table 8
Performance of the LLM4M traditional optimization model (based on the MSE, RMSE, MAE, and MAPE between the generated implementable plans and the optimal plan costs).

Model	MSE	RMSE	MAE	MAPE
LLM4M	574 629.5433	758.0432	453.1927	0.0200
GA	3 565 545.5720	1 888.2652	1 539.7948	0.0660
PSO	3 820 743.8065	1 954.6723	1 611.5536	0.0689
ACO	275 041.5270	524.4440	311.6987	0.0142
SA	1 377 198.4179	1 173.5410	812.1318	0.0356

Table 9
Detailed information of the datasets in different scenarios, including the number of maintenance teams, the number of nodes, and the number of training and test sets.

Dataset name	Number of maintenance team	Number of turbines required maintenance	Train data volume	Test data volume
TRFR	[2,10]	[3,20]	–	2 000
T3F5	3	5	10 000	1 000
T2F5	2	5	10 000	1 000
T5F10	5	10	10 000	1 000
T3F9	3	9	10 000	1 000
T3F10	3	10	10 000	1 000
T6F13	6	13	10 000	1 000
T4F11	4	11	10 000	1 000
T2F9	2	9	10 000	1 000
T2F10	2	10	10 000	1 000
T3F11	3	11	10 000	1 000
T5F6	5	6	10 000	1 000
T2F11	2	11	10 000	1 000
T2F12	2	12	10 000	1 000
T4F17	4	17	10 000	1 000

ations. Moreover, the constraint that each maintenance team must participate in the task is not explicitly stated in the prompt. LLM4M appears to have recognized this latitude and generated lower-cost plans by assigning fewer teams to the task than are assigned in the optimal plan. The data points in Fig. 12 with values less than zero directly result from this behavior. As shown in Fig. 13 and Table 11, the models trained on T5F10, T6F13, T4F11, and T5F16 demonstrate superior performance in terms of the MSE, RMSE, MAE, and MAPE.

To further investigate the factors influencing the generalization ability of LLM4M, training and evaluation are conducted on a range of datasets. As shown in Fig. 14, when the model is trained and tested on the same dataset (i.e., a fixed scenario), it generally has a strong ability to generate implementable plans. Furthermore, the model appears to capture the semantic context of scenarios similar to those seen during training. For instance, the model trained on the T3F5 dataset successfully generalizes to the scenario represented by T2F5 but produces largely invalid plans for the scenario represented by T4F17. Similarly, the model trained on T3F10 performs well on T2F9 and T2F10 but fails to interpret the scenario associated with T4F17. In addition, consistent with the findings from testing on the TRFR dataset, training on datasets that reflect more complex scenarios tends to increase the likelihood of hallucinations, resulting in incorrect outputs, as seen with T5F16, T2F11, T2F12, and T4F17.

In terms of planning capability, as shown in Fig. 15, the models trained on datasets with only two maintenance teams (e.g., T2F5, T2F9, T2F11, and T2F12) perform well only in similar scenarios. In other scenarios, their outputs exhibit highly dispersed distributions, indicating poor generalization in planning. Models trained

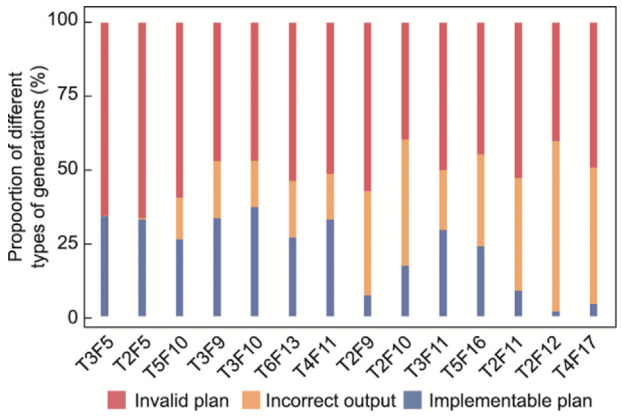

Fig. 11. Comparison of maintenance plan generation performance on the TRFR dataset. The *X*-axis represents the models trained on different datasets.

Table 10
Proportion of different generation types in the TRFR dataset.

Model	Implementable plan	Incorrect output	Invalid plan
T3F5	34.10%	0.15%	65.75%
T2F5	32.95%	0.50%	66.55%
T5F10	26.20%	14.20%	59.60%
T3F9	33.40%	19.35%	47.25%
T3F10	37.15%	15.70%	47.15%
T6F13	26.80%	19.30%	53.90%
T4F11	33.00%	15.45%	51.55%
T2F9	7.05%	35.65%	57.30%
T2F10	17.20%	42.95%	39.85%
T3F11	29.40%	20.30%	50.30%
T5F16	23.85%	31.25%	44.90%
T2F11	8.75%	38.40%	52.85%
T2F12	1.55%	58.00%	40.45%
T4F17	4.15%	46.35%	49.50%

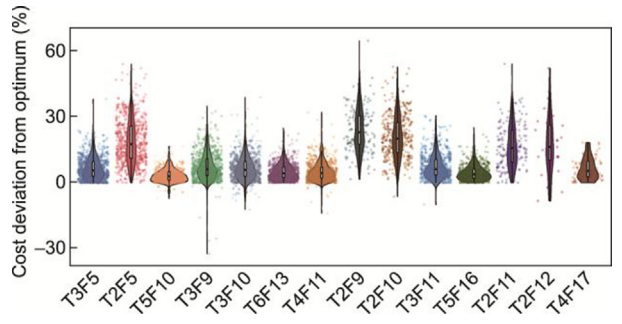

Fig. 12. Comparison of the cost deviations of the generated implementable on the TRFR dataset. The *X*-axis represents the models trained on different datasets.

on datasets from two or more maintenance teams demonstrated strong planning capabilities during testing, even in semantically simpler scenarios, such as T3F5 and T2F5. The MSE, RMSE, MAE, and MAPE values for LLM4M results on different test datasets are included in Appendix A Figs. S1 and S2.

Motivated by these phenomena, we aim to identify underlying patterns to inform best practices for future training dataset selection in this domain. Consequently, we systematically aggregated and analyzed test results from models trained on heterogeneous datasets, focusing on how the number of maintenance teams and fault scenarios in training samples influence the fine-tuning efficacy of LLM4M. By averaging the test metrics across diverse datasets and performing surface fitting on the aggregated results, we obtained the visualization presented in Fig. 16.

As illustrated in Fig. 16, the variation trends of different evaluation metrics across training datasets are substantially consistent, demonstrating the considerable impact of training dataset selection on the fine-tuning performance of LLM4M. To ensure visualization completeness, interpolation is applied to the raw data. Empirical results demonstrate that models trained on datasets from complex operational scenarios exhibit strong generalization capabilities to simpler environments, whereas the converse scenario—the application of models trained on simplified datasets to complex tasks—often fails to achieve comparable performance. Such findings underscore the paramount importance of adopting principled strategies for dataset complexity management in AI system development. Notably, the visualization reveals a distinct phase transition boundary that demarcates a sudden performance increase. To clarify this critical phenomenon, the interpolated three-dimensional surface is projected onto a two-dimensional plane in Fig. 17.

As depicted in Fig. 17, two distinct phase transition boundaries emerge at approximately three and five maintenance teams in the training scenarios, with the most pronounced performance inflection occurring around the threshold of three teams. This phenomenon mirrors phase transition behaviors observed in complex systems research, where systemic properties undergo qualitative transformations through collective dynamics upon crossing critical thresholds. When the number of maintenance teams is less than three, the system operates in a low-dimensional regime characterized by limited interaction combinations and simplistic task allocation patterns. Under these conditions, LLMs struggle to abstract universalizable rules from sparse training examples. However, when the number exceeds three, the system transitions to an edge-of-chaos regime in a high-dimensional space. Here, nonlinear relationships such as task conflicts, resource competition, and spatiotemporal coupling proliferate exponentially. For example, configurations with four teams may exhibit emergent behaviors such as dynamic team clustering and nested priority hierarchies. This complexity compels the model to develop sophisticated coordination strategies encompassing game-theoretic equilibria and adaptive priority scheduling mechanisms.

This observation has profound methodological implications: the strategic selection of training datasets should follow principled design rather than arbitrary trial-and-error. The identified team-count threshold demonstrates how seemingly minor parameter variations can induce fundamental regime shifts in model behavior, underscoring the need for systematic dataset curation frameworks in operational research applications.

4.4. Performance results of dynamic maintenance tasks

4.4.1. Description and dataset of nonfixed starting points

In real-world maintenance scenarios, to respond to real-time maintenance needs, maintenance teams may need to replan their operations during the maintenance process to address the latest maintenance requirements. To simulate this scenario, we no longer use a fixed starting point for the maintenance teams but instead randomize their starting points to different turbine sites to simulate the real-time updating of maintenance plans.

Therefore, on the basis of the above assumptions, we generated a dataset with 10 000 training samples and 1 000 test samples in the same manner as in Section 4.2. In this dataset, the starting points of the maintenance teams are not fixed, but all maintenance teams return to the origin. On the basis of this dataset, we set the following three scenes and conducted experiments:

Type 1: In the training and test data, maintenance teams start from the origin.

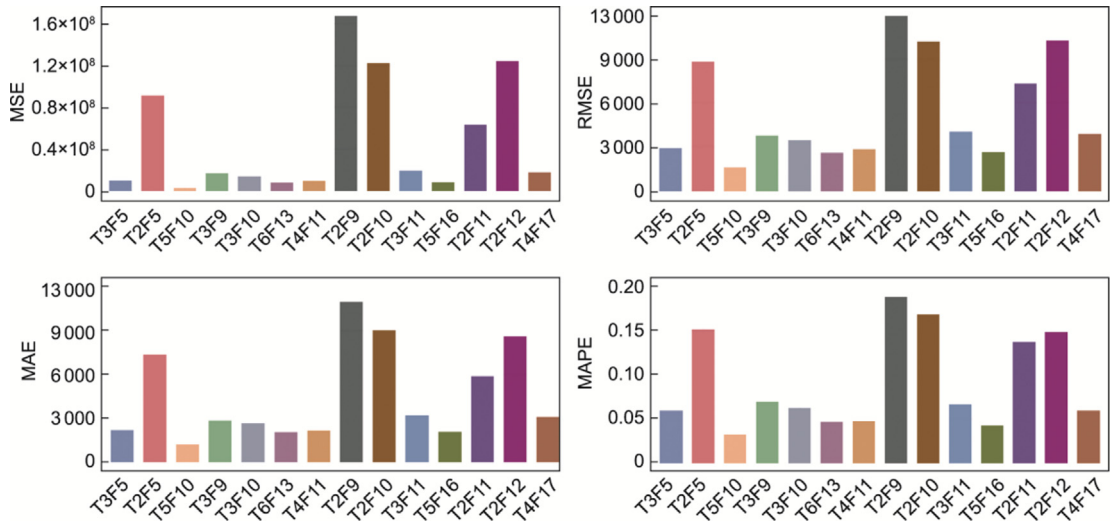

Fig. 13. Comparison of performance on the TRFR dataset (based on the MSE, RMSE, MAE, and MAPE between generated implementable plans and the optimal plan costs). The *X*-axis represents the models trained on different datasets.

Table 11
Performance on TRFR (based on the MSE, RMSE, MAE, and MAPE between generated implementable plans and the optimal plan costs).

Model	MSE	RMSE	MAE	MAPE
T3F5	9 928 745.2740	3 150.9910	2 384.6890	0.0600
T2F5	91 342 227.5470	9 557.3130	7 989.3120	0.1520
T5F10	2 982 151.7000	1 726.8910	1 321.6920	0.0330
T3F9	16 683 240.6010	4 084.5120	3 077.5170	0.0700
T3F10	13 928 637.2940	3 732.1090	2 895.0710	0.0630
T6F13	7 973 613.2260	2 823.7590	2 234.9120	0.0470
T4F11	9 472 273.9960	3 077.7060	2 343.6320	0.0480
T2F9	167 330 852.1740	12 935.6430	11 860.4460	0.1890
T2F10	122 289 360.6850	11 058.4520	9 762.6530	0.1690
T3F11	19 289 533.2480	4 391.9850	3 472.2830	0.0670
T5F16	8 160 259.5950	2 856.6170	2 251.1450	0.0430
T2F11	63 254 630.2400	7 953.2780	6 390.3060	0.1380
T2F12	124 212 859.5030	11 145.0820	9 339.1130	0.1490
T4F17	17 791 113.1080	4 217.9510	3 355.8550	0.0600

Type 2: In the training data, maintenance teams start from the origin, whereas in the test data, the teams are initially located at randomly selected turbine sites instead of the origin.

Type 3: In the training and test data, maintenance teams are initially located at randomly selected turbine sites.

4.4.2. Generation performance of nonfixed starting points scenarios

Fig. 18 and Table 12 show the final generation performance from Type 1 to Type 3. The Type 1 test results achieved 100% generation accuracy. In contrast, the Type 2 and Type 3 test results, while only 97.90% and 99.40%, respectively, also demonstrated excellent generation performance.

Fig. 19, Fig. 20, and Table 13 show the planning capabilities of the model under different scenes. In terms of test sample characteristics, we compared Type 1 and Type 2 models, fine-tuned them on the same training dataset, the T3F5 dataset, and tested their performance in static and real-time maintenance scenarios. Our experimental results revealed that although the Type 2 model had lower maintenance strategy generation and planning capabilities than the Type 1 model did, its mean absolute error (MAPE) remained at 0.1511, indicating that the performance was acceptable.

Furthermore, we compared Type 2 with Type 3 in terms of the model's dynamic planning capability. The training data for Type 2 did not have dynamic planning capability, whereas those for Type 3 did. The results demonstrated that Type 3 outperforms Type 2 across all the evaluation metrics, reflecting generation and decision-making capabilities. This comparison suggests that a model whose training data include dynamic planning capability is better equipped to comprehend and process dynamic planning data, thereby significantly enhancing its overall performance.

4.5. Model inference cost and economic consumption

When the practical deployment feasibility of LLMs in industrial settings is evaluated, beyond their core generative and planning capabilities, the long-term operational cost and inference latency are equally critical considerations.

The operational costs of a model primarily fall into two scenarios: on-premises deployment and API calls. For models requiring on-premises deployment, such as the LLM4M discussed in this paper, the cost is primarily an upfront investment in computing hardware. Our fine-tuning process, which was based on the Qwen2.5-7B foundational model, was conducted using two 40 G A100 GPUs. Notably, for devices equipped with approximately 50–60 GB of video random access memory (VRAM), the model can also be fine-tuned within an acceptable timeframe. Depending on the complexity of the model, the fine-tuning time is generally 40 to 150 min. Furthermore, during the practical usage phase (i.e., the inference stage), equally high-cost hardware is not needed. Some consumer-grade GPUs are sufficient to run the model smoothly, which considerably reduces the actual

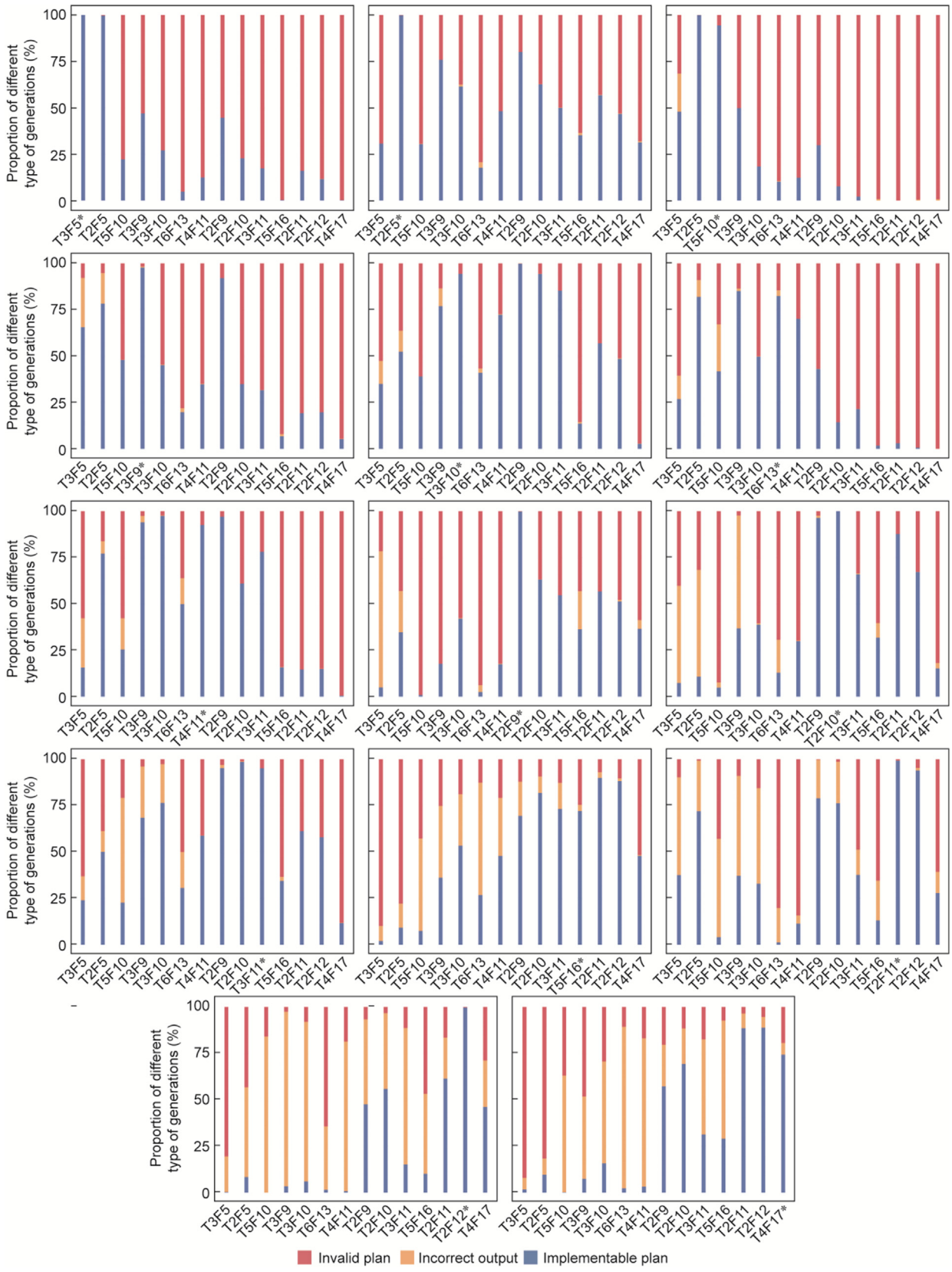

Fig. 14. Comparison of maintenance plan generation performance on different datasets. The *X*-axis represents the testing datasets, and * indicates that the results shown are from the model trained on the corresponding dataset.

deployment costs. In contrast, the cost of using an API is contingent on the pricing strategies of different service providers. The approximate pricing from major LLM vendors is as follows:

GPT: The input pricing is approximately 1.25–2.5 USD per million tokens, and the output pricing is approximately 10–12 USD per million tokens [49].

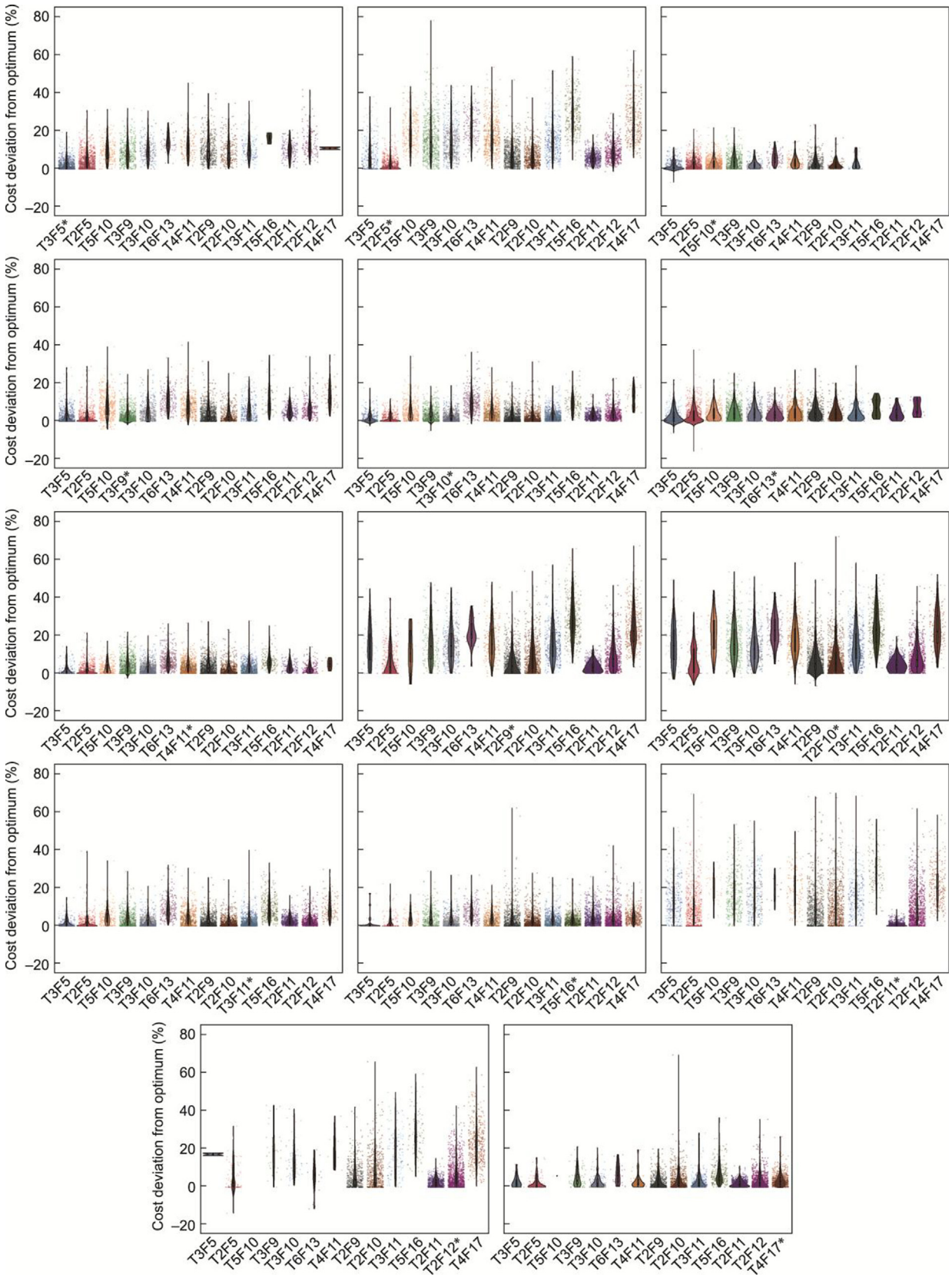

Fig. 15. Comparison of cost deviations generated on different datasets. The *X*-axis represents the testing datasets, and * indicates that the results shown are from the model trained on the corresponding dataset.

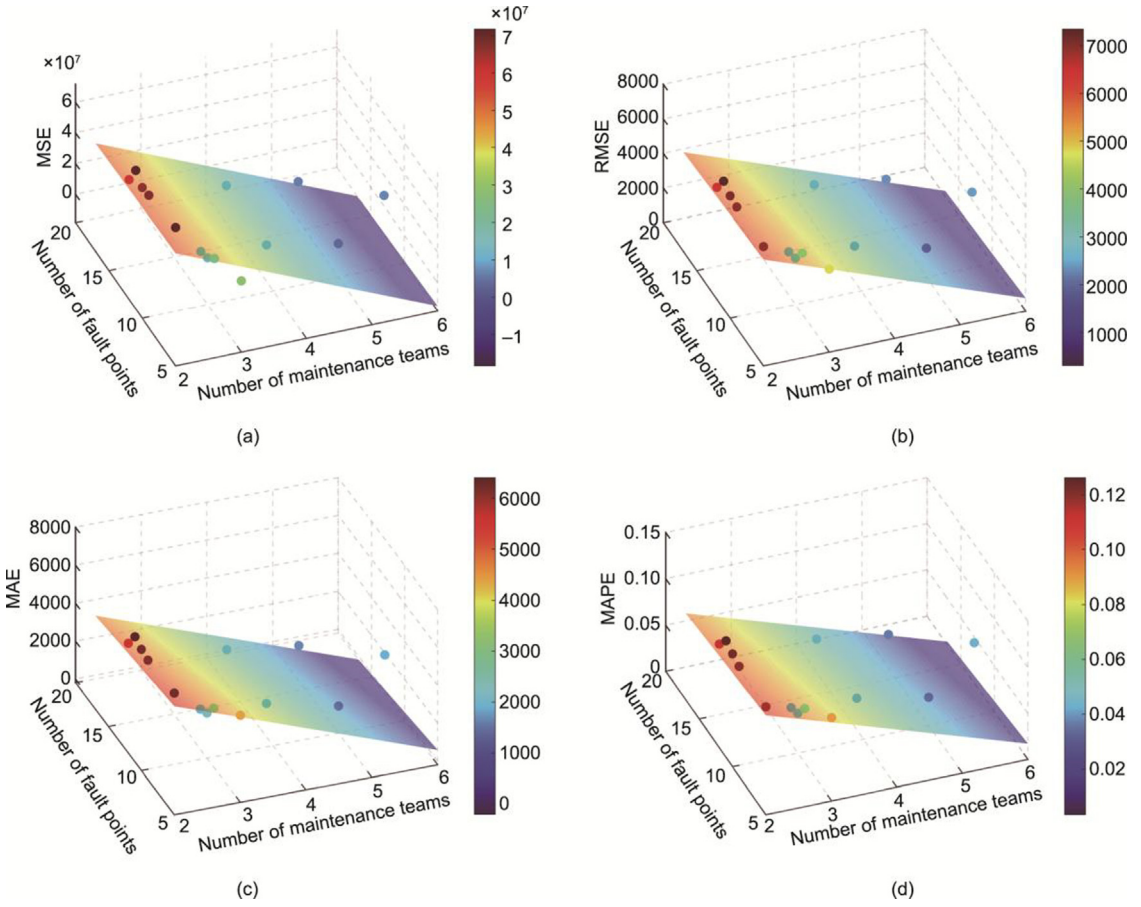

Fig. 16. Performance comparison of different training datasets on the test set across key metrics. (a) MSE; (b) RMSE; (c) MAE; (d) MAPE.

Claude: The input pricing is approximately 1–3 USD per million tokens, and the output pricing is approximately 5–15 USD per million tokens [50].

DeepSeek: The pricing is relatively uniform, with input and output prices of 0.56 USD per million tokens and 1.68 USD per million tokens, respectively [51].

Qwen: The input pricing is approximately 2–4 USD per million tokens, and the output pricing is approximately 8–12 USD per million tokens [52].

The main factors affecting LLM output speed include the number of tokens and network quality. With respect to the number of tokens, we selected three test scenarios—T3F5, T2F10, and T4F17—with a small, moderate, and large number of tokens, respectively. The average inference times are detailed in Table 14. Notably, although LLM4M was run locally, owing to the limited hardware used in the experiment, its inference speed may be inferior to that of other LLMs that provide cloud services. However, this inference speed can still be considered an effective and fast response to users. The latter benefit from extensive backend infrastructure, which allows them to offer significantly faster response times to users. From the perspective of network quality, the use of APIs requires a stable network connection, which incurs additional call costs and connection pressure. LLM4M, with the advantage of local deployment, does not have additional call costs or connection pressure.

5. Conclusions and further work

In this study, a domain-specific LLM-enabled approach for maintenance decision-making for wind farms with diverse numbers of wind turbines and maintenance teams is proposed. An advanced LLM for maintenance (LLM4M) has been established and trained on extensive datasets of maintenance decision-making for wind farms generated from the MILP results. Given inputs such as the layout of turbines, failure modes, number of maintenance teams, and the cost and time for the maintenance team to accomplish the repair, LLM4M can accurately generate the task allocation for the maintenance teams and optimal maintenance route planning in a short time, considerably reducing the difficulty and time cost of modeling compared with conventional MILP modeling and solutions.

The conclusions drawn from this study are as follows:

- **Model validity and accuracy**: The proposed fine-tuned LLM4M achieved a 100% effective probability of generating task allocation for maintenance teams and practical maintenance route planning. Compared with non-fine-tuned LLM models, the proposed LLM4M obtained 2.00% MAPE for the optimal results, while the Claude, DeepSeek, GPT, and Qwen LLMs obtained 13.78%, 15.26%, 14.85%, 14.24%, respectively.
- **Model generalization**: The proposed LLM4M model exhibits promising generalization capabilities. The fine-tuned LLM4M model achieves competitive performance on the TRFR benchmark, except for models trained with the T2F5, T2F9, T2F10, T2F11, and T2F12 datasets. These findings warrant attention, as the underperforming models were exclusively trained on maintenance data from two maintenance teams. The observed performance disparity suggests that the LLM4M model may demonstrate sensitivity to specific characteristics of training data sources, potentially related to variations in maintenance teams. Moreover, the results of dynamic maintenance tasks

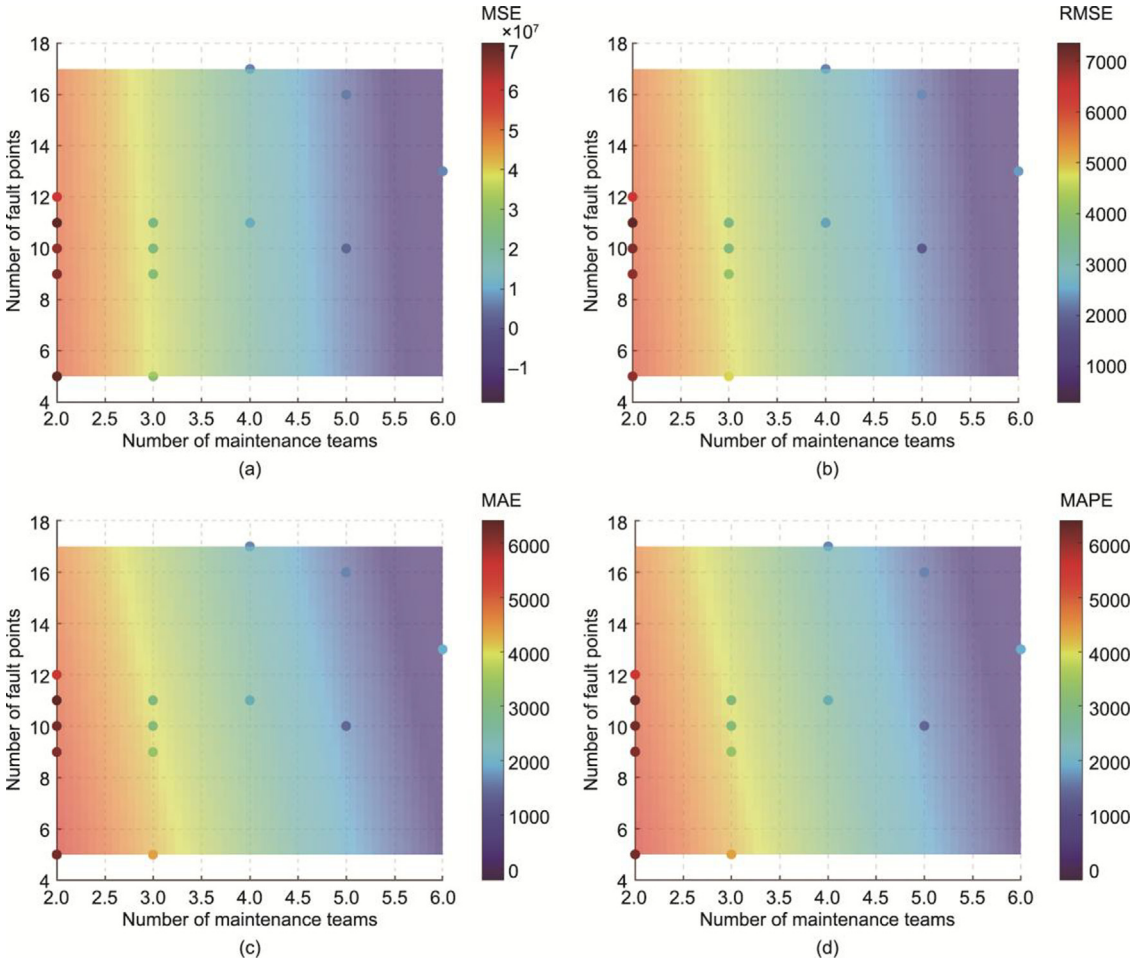

Fig. 17. Performance landscape analysis: impact of training datasets on key metrics. (a) MSE; (b) RMSE; (c) MAE; (d) MAPE.

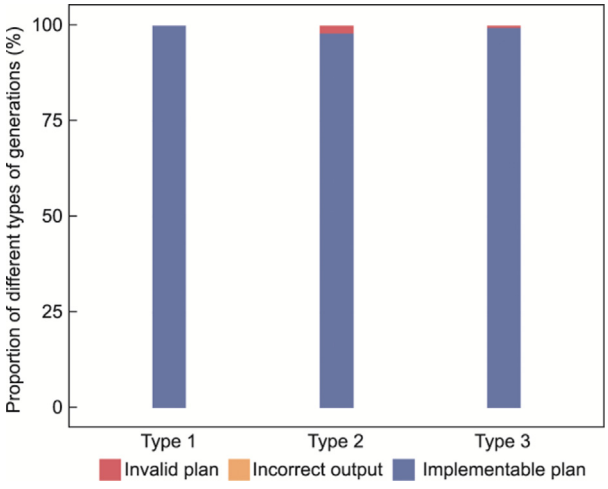

Fig. 18. Comparison of maintenance plan generation performance for Types 1, 2, and 3.

Table 12
Proportion of different generation from Types 1, 2, and 3.

Type	Implementable plan	Incorrect output	Invalid plan
Type 1	100%	0	0
Type 2	97.90%	0	2.10%
Type 3	99.40%	0	0.60%

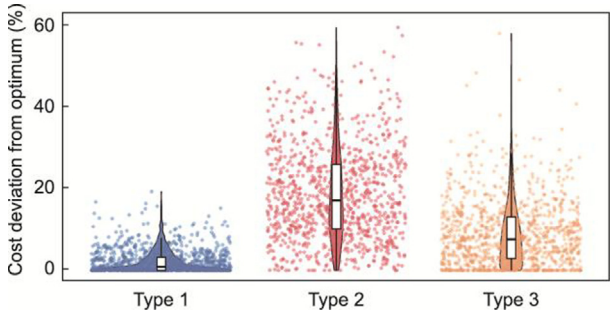

Fig. 19. Comparison of cost deviations within implementable plans generated for Types 1, 2, and 3.

demonstrate that the LLM4M can effectively adapt to realistic dynamic task scenarios. This insight can further inform the design and curation of training datasets for different types of tasks in practical deployments, facilitating the development of models with improved generalization performance. Nonetheless, the proposed model holds considerable practical value, as conventional analytical and ML models usually require several hours to solve such problems—depending on the size of the solution space—whereas the proposed model can generate implementable maintenance plans within seconds once trained. In addition, compared with major LLMs, LLM4M features lower deployment costs and functions independently of internet access.

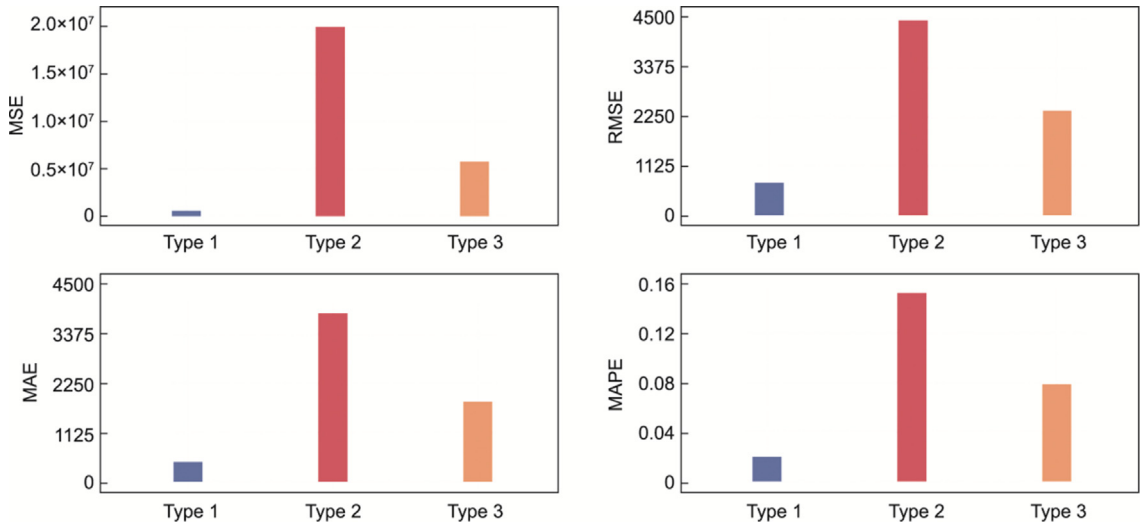

Fig. 20. Comparison of the performance of Types 1, 2, and 3 (based on the MSEs, RMSEs, MAEs, and MAPEs between the generated implementable plans and the optimal plan costs).

Table 13
Performances of Types 1, 2 and 3 (based on the MSEs, RMSEs, MAEs, and MAPEs between the generated implementable plans and the optimal plan costs).

Type	MSE	RMSE	MAE	MAPE
Type 1	**574 629.5433**	**758.0432**	**453.1927**	**0.0200**
Type 2	19 737 941.4863	4 442.7403	3 814.5749	0.1511
Type 3	5 773 766.3022	2 402.8663	1 828.8961	0.0786

Table 14
Average inference times for LLM4M and LLMs.

Test scenarios	Average inference time (s)				
	LLM4M	Claude	DeepSeek	GPT	Qwen
T3F5	7.11	1.54	3.34	0.79	0.76
T2F10	9.35	1.63	3.56	1.19	1.14
T4F17	17.79	2.25	4.65	1.36	1.28

- **Model characteristics**: A phase transition behavior was observed. This behavior indicated that systemic properties undergo qualitative transformations through collective dynamics upon crossing critical thresholds. When the number of maintenance teams exceeds a specific value, the system transitions to an edge-of-chaos regime in a high-dimensional space. This finding provides experience and guidance for relevant scholars in the model training of maintenance decision-making problems.

To enhance the applicability of this approach to maintenance planning, future work will focus on expanding the dataset by incorporating a broader range of more complex maintenance models and parameters, such as maintenance capabilities, weather factors and other limitations. Moreover, preventive and predictive maintenance processes should be incorporated into the LLM model. Uncertainty in decision-making is also a key focus of maintenance. Increasing the input channel features in the LLM model to accommodate these additional parameters will enable a more comprehensive representation of maintenance decision-making, enhancing its ability to generalize across diverse maintenance scenarios. Explainability, as another crucial future direction, is essential for building trust and promoting the practical adoption of AI in engineering. Future research will explore and develop explainability methods to better understand the model's decision-making process. Furthermore, integrating physics-based constraints into the model training process can be explored in future studies to enhance the reliability of the AI-generated outputs.

CRediT authorship contribution statement

Dongming Fan: Writing – review & editing, Writing – original draft, Validation, Funding acquisition, Conceptualization. **Meng Liu:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Yi Shao:** Writing – review & editing, Writing – original draft, Software, Methodology. **Linchao Yang:** Writing – original draft, Visualization, Supervision, Project administration, Funding acquisition, Conceptualization. **Yiliu Liu:** Writing – original draft, Validation, Supervision. **Yue Zhang:** Investigation, Data curation. **Yi Ren:** Validation, Supervision, Project administration. **Zili Wang:** Validation, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for this journal and was not involved in the editorial review or the decision to publish this article.

Acknowledgments

This research was funded by the National Natural Science Foundation of China (72401097, 72301016, and 72571015), the Beijing Nova Program, and the Fundamental Research Funds for the Central Universities.

Declaration of generative AI in scientific writing

During the preparation of this work, the authors used GPT-4o and GPT-5 to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Appendix A. Supplementary Data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eng.2025.12.019.

References

[1] Bai X, Fan Y, Hou J, Sun Y, Liu Y, Liu J. Reliability evaluation of direct current gathering system in onshore wind farm based on reliability block diagram-sequential monte carlo. Sustain Energy Grids Netw 2024;40:101549.
[2] Tao Z, Liu H, Si Y, Wang C, Zhu R. An opportunistic joint maintenance strategy for two offshore wind farms. Ocean Eng 2024;304:117890.
[3] Elsahhar AU, Ezzat AA, Elsabbagh A, Elbanhawy AY. Analysis of failure and maintenance records in aging wind farms to inform end-of-life asset management. Wind Energy 2025;28(6):e70024.
[4] Tao Z, Zhu R, Hu J, Wang M, Chen Q, Wang C. A novel hierarchical failure analysis approach targeting the operation and maintenance of floating offshore wind turbines. Renew Energy 2025;241:122267.
[5] Izquierdo J, Márquez AC, Uribetxebarria J, Erguido A. On the importance of assessing the operational context impact on maintenance management for life cycle cost of wind energy projects. Renew Energy 2020;153:1100–10.
[6] Wang H, Li YF, Ren J. Machine learning for fault diagnosis of high-speed train traction systems: a review. Front Eng Manag 2024;11:62–78.
[7] Pelajo JC, Brandão LET, Gomes LL, Klotzle MC. Wind farm generation forecast and optimal maintenance schedule model. Wind Energy 2019;22:1872–90.
[8] Dui H, Wu X, Wu S, Xie M. Importance measure-based maintenance strategy optimization: fundamentals, applications and future directions in AI and IoT. Front Eng Manag 2024;11:542–67.
[9] Niemi A, Skobiej B, Kulev N, Sill TF. Modeling offshore wind farm disturbances and maintenance service responses within the scope of resilience. Reliab Eng Syst Saf 2024;242:109719.
[10] Ade Irawan C, Starita S, Chan HK, Eskandarpour M, Reihaneh M. Routing in offshore wind farms: a multi-period location and maintenance problem with joint use of a service operation vessel and a safe transfer boat. Eur J Oper Res 2023;307:328–50.
[11] Fan D, Sun B, Dui H, Zhong J, Wang Z, Ren Y, et al. A modified connectivity link addition strategy to improve the resilience of multiplex networks against attacks. Reliab Eng Syst Saf 2022;221:108294.
[12] Su C, Wu L. Opportunistic maintenance optimisation for offshore wind farm with considering random wind speed. Int J Prod Res 2024;62:1862–78.
[13] Li M, Bijvoet B, Wu K, Jiang X, Negenborn RR. Optimal chartering decisions for vessel fleet to support offshore wind farm maintenance operations. Ocean Eng 2024;298:117202.
[14] Wang Y, Han Y, Gong D, Li H. A review of intelligent optimization for group scheduling problems in cellular manufacturing. Front Eng Manag 2023;10:406–26.
[15] Fan D, Ren Y, Feng Q, Liu Y, Wang Z, Lin J. Restoration of smart grids: current status, challenges, and opportunities. Renew Sustain Energy Rev 2021;143:110909.
[16] Zhao X, Chen P, Tang LC. Condition-based maintenance via Markov decision processes: a review. Front Eng Manag 2025;12:330–42.
[17] Qureshi TA, Warudkar V. Wind farm optimization by active yaw control strategy using machine learning approach. Int J Green Energy 2024;21:2628–39.
[18] M'zoughi F, Lekube J, Garrido AJ, De La Sen M, Garrido I. Machine learning-based diagnosis in wave power plants for cost reduction using real measured experimental data: Mutriku Wave Power Plant. Ocean Eng 2024;293:116619.
[19] Ahmed SF, Alam MSB, Hassan M, Rozbu MR, Ishtiak T, Rafa N, et al. Deep learning modeling techniques: current progress, applications, advantages, and challenges. Artif Intell Rev 2023;56:13521–617.
[20] Lee N, Woo J, Kim S. A deep reinforcement learning ensemble for maintenance scheduling in offshore wind farms. Appl Energy 2025;377:124431.
[21] Kavousi-Fard A, Dabbaghjamanesh M, Sheikh M, Jin T. A novel deep learning based digital twin model for mitigating wake effects in wind farms. Renew Energy Focus 2025;53:100686.
[22] Wang Y, Zhao X, Li Z, Zhu W, Gui R. A novel hybrid model for multi-step-ahead forecasting of wind speed based on univariate data feature enhancement. Energy 2024;312:133515.
[23] Wu F, Shen T, Bäck T, Chen J, Huang G, Jin Y, et al. Knowledge-empowered, collaborative, and co-evolving AI models: the post-LLM roadmap. Engineering 2025;44:87–100.
[24] Fu T, Liu S, Li P. Intelligent smelting process, management system: efficient and intelligent management strategy by incorporating large language model. Front Eng Manag 2024;11:396–412.
[25] Zhang J, Zhang C, Lu J, Zhao Y. Domain-specific large language models for fault diagnosis of heating, ventilation, and air conditioning systems by labeled-data-supervised fine-tuning. Appl Energy 2025;377:124378.
[26] Liu Y, Zhou Y, Liu Y, Xu Z, He Y. Intelligent fault diagnosis for CNC through the integration of large language models and domain knowledge graphs. Engineering 2025;53:311–22.
[27] Cui H, Guo X, Yu L. Leveraging pre-trained GPT models for equipment remaining useful life prognostics. Electronics 2025;14(7):1265.
[28] Ye W, Zhang Z, Li W, Qin Y, Zhang P. CRALA: a Chinese-centric risk simulation and assessment framework for LLM agents. In: Proceedings of the 2024 4th International Symposium on Artificial Intelligence and Intelligent Manufacturing (AIIM); 2024 Dec 20–22; Chengdu, China. New York City: IEEE; 2024. p. 561–6.
[29] Addison L, Hosang A, Tuitt TA, Manohar K, Hosein P. A LLM-based platform for flood risk education and weather alerts in SIDS. In: Proceedings of the 2024 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD); 2024 Nov 4–6; University City, Sharjah. New York City: IEEE; 2024. p. 1–6.
[30] Hu J, Li Q, Fu N. Generative AI for materials discovery: design without understanding. Engineering 2024;39:13–7.
[31] You Y, Tan K, Jiang Z, Zhang L. Developing a predictive platform for salmonella antimicrobial resistance based on a large language model and quantum computing. Engineering 2025;48:174–84.
[32] Wang H, Gao C, Dantona C, Hull B, Sun J. DRG-LLaMA: tuning LLaMA model to predict diagnosis-related group for hospitalized patients. npj Digit Med 2024;7:16.
[33] Liu Y, Li X, Luo Y, Du J, Zhang Y, Lv T, et al. Toward a large language model-driven medical knowledge retrieval and QA system: framework design and evaluation. Engineering 2025;50:270–82.
[34] Zhang J, Li Y, Liu Y, Xie L. Research and practice of NL2SQL technology based on LLM for big data of enterprise finance. In: Proceedings of the 2024 4th International Conference on Advanced Enterprise Information System (AEIS); 2024 Dec 6–8; Cambridge, UK. New York City: IEEE; 2024. p. 46–51.
[35] Wei F, Keeling R, Huber-Fliflet N, Zhang J, Dabrowski A, Yang J, et al. Empirical study of LLM fine-tuning for text classification in legal document review. In: Proceedings of the 2023 IEEE International Conference on Big Data (BigData); 2023 Dec 15–18; Sorrento, Italy. New York City: IEEE; 2023. p. 2786–92.
[36] Bock S, Bomsdorf S, Boysen N, Schneider M. A survey on the Traveling Salesman Problem and its variants in a warehousing context. Eur J Oper Res 2025;322:1–14.
[37] Liu M, Feng Q, Fan D, Dui H, Sun B, Ren Y, et al. Resilience importance measure and optimization considering the stepwise recovery of system performance. IEEE Trans Reliab 2023;72:1064–77.
[38] Fan D, Feng Q, Zhang A, Liu M, Ren Y, Wang Y. Optimization of scheduling and timetabling for multiple electric bus lines considering nonlinear energy consumption model. IEEE Trans Intell Transport Syst 2024;25:5342–55.
[39] Sun C, Huang S, Pompili D. LLM-based multi-agent decision-making: challenges and future directions. IEEE Robot Autom Lett 2025;10(6):5681–8.
[40] Lin Z. How to write effective prompts for large language models. Nat Hum Behav 2024;8:611–5.
[41] Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput Surv 2023;55(9):195:1–35.
[42] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al. LoRA: low-rank adaptation of large language models. arXiv.2106.09685.
[43] Yang A, Yang B, Zhang B, Hui B, Zheng B, Yu B, et al. Qwen2.5 technical report 2025. arXiv.2412.15115.
[44] Han L, Zhang Y, Gao W, Li X. LexSage: multi-task optimization in legal large language model applications. In: Proceedings of the 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC); 2024 Dec 27–29; Xiamen, China. New York City: IEEE; 2024. p. 325–30.
[45] Abacha AB, Yim W, Fu Y, Sun Z, Yetisgen M, Xia F, et al. MEDEC: a benchmark for medical error detection and correction in clinical notes. arXiv.2412.19260.
[46] Fan D, Ren Y, Feng Q, Zhu B, Liu Y, Wang Z. A hybrid heuristic optimization of maintenance routing and scheduling for offshore wind farms. J Loss Prev Process Ind 2019;62:103949.
[47] Zhang ZY. Scheduling and routing optimization of maintenance fleet for offshore wind farms using duo-ACO. Adv Mater Res 2014;1039:294–301.
[48] Fan D, Zhang A, Feng Q, Cai B, Liu Y, Ren Y. Group maintenance optimization of subsea Xmas trees with stochastic dependency. Reliab Eng Syst Saf 2021;209:107450.
[49] OpenAI. Pricing [Internet]. San Francisco: OpenAI; undated [cited 2025 Sep 9]. Available from: https://openai.com/api/pricing/.

[50] Anthropic PBC. Pricing [Internet]. San Francisco: Anthropic PBC; undated [cited 2025 Sep 9]. Available from: https://www.anthropic.com/pricing.
[51] DeepSeeks. Models & pricing [Internet]. Hangzhou: DeepSeeks; undated [cited 2025 Sep 9]. Available from: https://deepseeks.pro/models-and-pricing/#:~:text=DeepSeek%20AI%202024%20pricing%3A%20Compare%20Chat%20%26%20Reasoner,discounts%2C%20and%20API%20costs.%20Features%20%26%20savings%20guide.
[52] Alibaba Cloud. Alibaba cloud model studio [Internet] Hangzhou: Alibaba Cloud; undated [cited 2025 Sep 9]. Available from: https://www.alibabacloud.com/help/en/model-studio.