Journal of Zhejiang University-SCIENCE A
www.jzus.zju.edu.cn; www.springer.com/journal/11582
E-mail: jzus_a@zju.edu.cn

Research Article
https://doi.org/10.1631/jzus.A2500629

Multi-agent collaborative optimization framework for integrated energy system scheduling

Boan QU[1*], Haoyu JIANG[2*], Songjie WANG[2], Zheng LUO[2], Xueru LIN[2,3], Xingtao TIAN[4], Jian LI[5], Xiaojie LIN[2,6,7✉], Wei ZHONG[1,2]

[1]*Polytechnic Institute, Zhejiang University, Hangzhou 310015, China*
[2]*College of Energy Engineering, Zhejiang University, Hangzhou 310027, China*
[3]*School of Information and Electrical Engineering, Hangzhou City University, Hangzhou 310015, China*
[4]*Key Laboratory of Cleaner Intelligent Control on Coal & Electricity, Ministry of Education, Taiyuan University of Technology, Taiyuan 030024, China*
[5]*School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China*
[6]*Shanghai Institute for Advanced Study, Zhejiang University, Shanghai 201203, China*
[7]*Changzhou Industrial Technology Research Institute, Zhejiang University, Changzhou 213022, China*

Abstract: Integrated energy system (IES) scheduling optimization involves multi-energy coupling, multidevice coordination, and complex constraints. Conventional approaches rely on experts to manually formulate optimization models, resulting in high technical barriers, lengthy development cycles, and limited transferability. To address requirement uncertainty, attention drift in long contexts, and high debugging costs when applying large language models (LLMs) to IES scheduling optimization, this study proposes the MASEO, a multi-agent system for energy optimization. MASEO decomposes the optimization workflow into four sequential stages—information collection, mathematical modeling, code implementation, and execution verification—each handled by a dedicated LLM agent. A structured checklist is introduced to standardize information collection, while an error-classification-based traceable repair mechanism routes errors to the responsible agent for targeted correction. Case studies on a Beijing data center IES and a German community IES show that MASEO achieves results highly consistent with expert benchmarks, with an annualized cost deviation of approximately 1.4% in the Beijing case. Compared with a single-agent baseline, modeling accuracy improves by approximately 35%, while the code execution pass rate increases from 63% to 94%. Ablation, multi-backbone, and sensitivity experiments further validate the effectiveness of the core mechanisms and the applicability of the framework. These results indicate that MASEO can reduce reliance on specialized modeling expertise and provide a practical pathway for the reliable application of LLMs to energy system optimization.

Key words: Integrated energy systems (IESs); Optimal scheduling; Large language models (LLMs); Multi-agent systems

1 Introduction

1.1 Background and motivation

Against the backdrop of the global energy transition and carbon neutrality goals, integrated energy systems (IESs) have emerged as a key pathway for improving energy efficiency through synergistic optimization of electricity, heat, cooling, and gas, offering notable advantages in economics, reliability, and environmental performance (Liu et al., 2014; Song et al., 2022; Huang et al., 2023).

However, IES scheduling optimization is highly complex due to multi-energy coupling, multidevice coordination, and diverse constraints (Fig. 1). Significant scenario heterogeneity exists across regions, and the problem demands interdisciplinary expertise spanning energy engineering, operation research, and computer science. Existing methods rely on experts to manually formulate optimization models and solve them using frameworks such as Oemof and PyPSA, suffering from high technical barriers, time-consuming development cycles, and limited transferability (Fan et al.,

✉ Xiaojie LIN, xiaojie.lin@zju.edu.cn
* The two authors contributed equally to this work
Xiaojie LIN, https://orcid.org/0000-0002-0829-1143

Received Nov. 30, 2025; Revision accepted June 10, 2026; Crosschecked Aug. 12, 2026

© Zhejiang University Press 2026

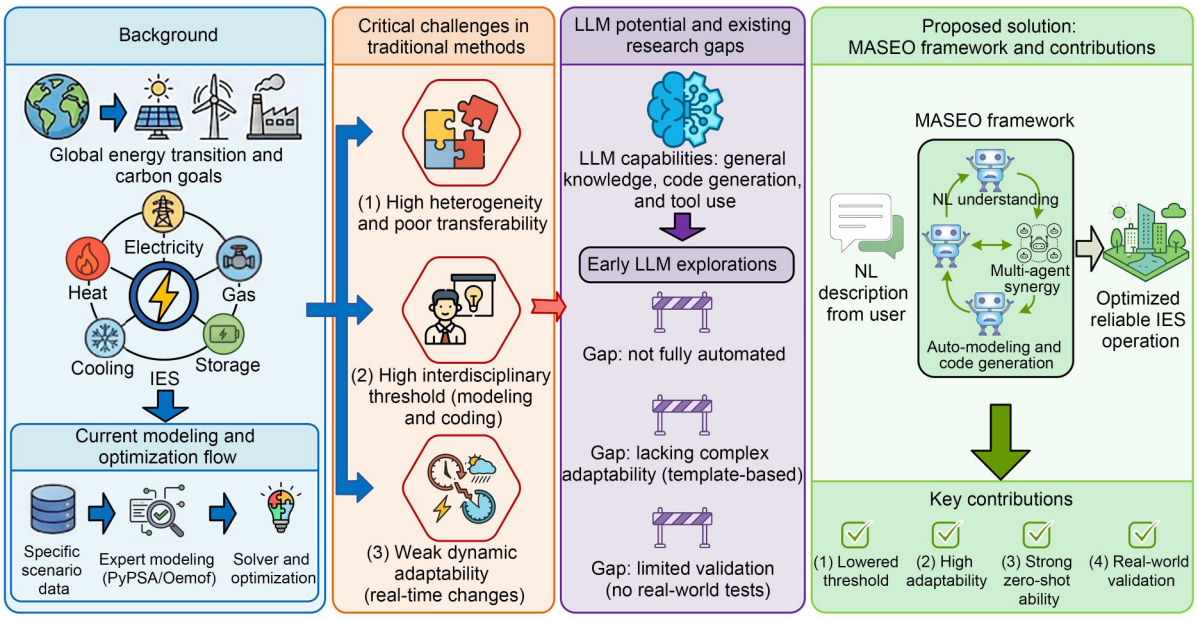


Fig. 1 Limitations of current scheduling methods for IES. NL is the natural language

2022; Majidi et al., 2025). Additionally, the dynamic operating environment of IES can invalidate existing model assumptions, while static modeling approaches struggle to adapt rapidly.

Recent advances in large language model (LLM)-based agent technologies offer a promising alternative (Lin et al., 2025). LLMs possess natural language understanding, code generation (Li et al., 2023; Roziere et al., 2023), and external tool invocation capabilities (Du et al., 2024; Zeng et al., 2024), enabling natural language-driven model construction, and thus reducing the dependence on specialized expertise (Jiang HY et al., 2026a, 2026b; Jiang JY et al., 2026). However, challenges still remain, including requirement uncertainty from natural language ambiguity, attention drift in long contexts, high debugging costs, and insufficient process transparency.

To address these issues, this study proposes the MASEO, a multi-agent system for energy optimization. The framework decomposes scheduling optimization into four stages—information collection, mathematical modeling, code implementation, and execution verification—completed by specialized agents to reduce the single-process information load and improve task focus and traceability. A guided interaction mechanism ensures complete and accurate information collection, while a traceable local repair mechanism classifies errors and routes them to upstream agents for targeted correction.

1.2 Literature review and research gap

In the domain of traditional optimization modeling and solving, extensive research has explored scheduling optimization across diverse IES configurations, spanning industrial parks, data centers, and community energy systems (Yang YF et al., 2024; Zhu et al., 2025). The emergence of open-source frameworks, such as Oemof and PyPSA, has improved modeling efficiency and reduced manual programming effort by providing modular component support (Hilpert et al., 2018; di Bella et al., 2024). However, given the significant heterogeneity among IES in terms of device composition, energy topology, and operational constraints, optimization models are highly scenario-specific and difficult to transfer across applications. Each new scenario requires domain experts to reconduct system analysis, model construction, and code implementation from scratch, resulting in high technical barriers and lengthy development cycles that constrain the broader adoption of optimization technologies across diverse energy system deployments.

In the domain of LLM-based optimization modeling and solving, research has pursued two directions. The first employs multi-agent systems to simulate heuristic optimization processes as a substitute for conventional solvers (Liu et al., 2024; Yang CR et al., 2024; Wang YD et al., 2025), but such approaches have been validated only on simple combinatorial problems such as

the traveling salesman problem and fall significantly short of mature solvers in solution accuracy, scalability, and real-time performance, rendering them unsuitable for engineering-grade IES dispatch. The second retains conventional solvers and leverages LLM capabilities of natural language understanding and code generation to assist with modeling—including optimization formalization from natural language descriptions (Jin et al., 2023), generation of power system simulation code (Jia et al., 2024), and LLM-based agents using the reasoning and acting (ReAct) paradigm for home energy management (Michelon et al., 2025)—reducing knowledge barriers and manual modeling workload to a certain extent. However, these efforts have focused on relatively simple, single-energy, or small-scale systems and have not been extended to engineering-grade IES scheduling involving multi-energy coupling and complex constraints (Fig. 2).

At a fundamental level, the widespread adoption of scheduling optimization technology faces a persistent core barrier: regardless of the tools employed, users must bear substantial learning and implementation costs for each new scenario—understanding problem structures, constructing mathematical models, and debugging optimization code. These demands on domain expertise and time investment prevent optimization technologies from reaching a broader range of users and heterogeneous energy system configurations (Jin et al., 2023; Michelon et al., 2025). While applying LLMs or single LLM-based agents to simple optimization tasks has demonstrated potential for lowering modeling barriers, the success rate remains low when facing IES scheduling—a multistage engineering problem involving multi-energy coupling and complex constraints—and users still require considerable time for manual debugging and intervention. How to construct a reliable end-to-end pipeline from requirement acquisition and mathematical modeling to code execution in a human–artificial intelligence (AI) collaborative manner remains an open and pressing challenge. To this end, this paper proposes the MASEO, addressing this challenge through multi-agent task decomposition, a structured checklist, as well as an error tracing and repair mechanism.

1.3 Contributions and study organization

(1) A multi-agent collaborative framework, MASEO, is proposed for IES scheduling optimization. Four specialized agents—problem formulation, model construction, code generation, and execution feedback—collaboratively transform natural language requirements into executable optimization workflows, addressing the complexity and expertise dependence of traditional approaches.

(2) A reliability assurance mechanism is proposed, incorporating a structured checklist for guided information collection and an error classification-based traceable repair mechanism for iterative correction. Physical feasibility is verified through energy balance checks and equipment boundary validation.

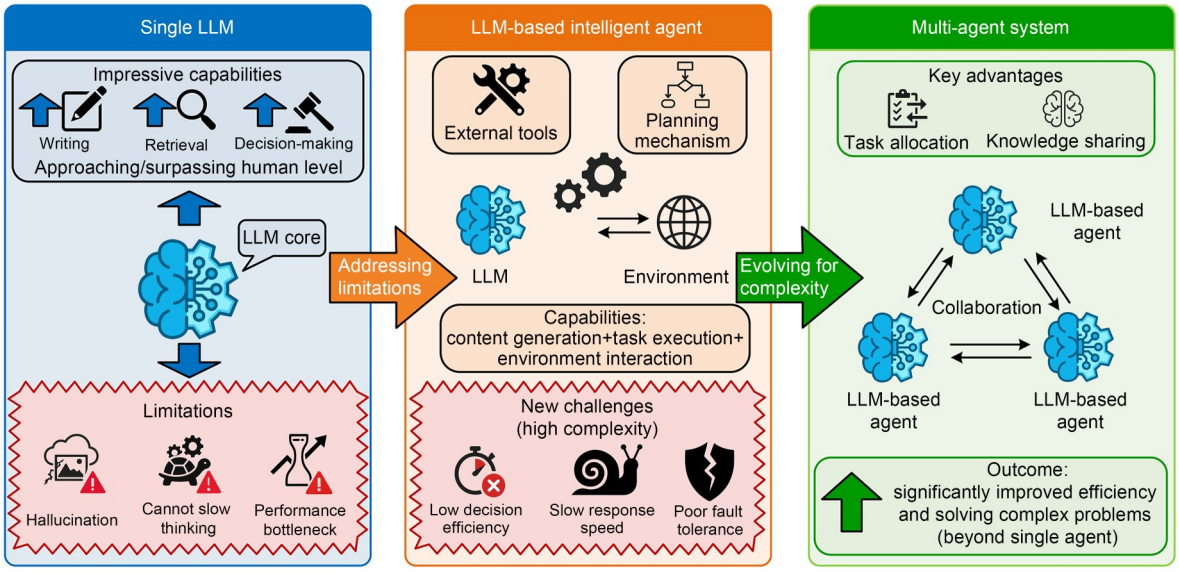


Fig. 2 Evolution of LLM-based approaches toward multi-agent systems

(3) Case studies on a Beijing data center IES (DCIES) and a German community IES (CIES) validate the framework. MASEO completes scheduling optimization at the hourly level through human–machine interaction, improving modeling accuracy by approximately 35% and the code execution pass rate from 63% to 94% over a single-agent approach. Multi-backbone and sensitivity experiments further confirm adaptability and provide configuration guidance for deployment.

2 Methodology

2.1 Basic definition of the IES optimization problem

A general optimization problem can be expressed in the following mathematical form:

$$\begin{cases} \min\limits_{x\in X} f(\boldsymbol{x};\boldsymbol{\theta}), \\ g_i(\boldsymbol{x};\boldsymbol{\theta})\leqslant 0,\ i=1,\ 2,\ \cdots,\ m, \\ h_j(\boldsymbol{x};\boldsymbol{\theta})=0,\ j=1,\ 2,\ \cdots,\ n, \end{cases} \tag{1}$$

where $\boldsymbol{x}\in\boldsymbol{X}\subseteq\mathbb{R}^n$ is the vector composed of all decision variables of the problem; $\boldsymbol{X}$ is the feasible region defined by physical constraints; $\boldsymbol{\theta}$ is the parameter set including objective function coefficients, constraint parameters, and system configuration information; $f(\boldsymbol{x};\boldsymbol{\theta})$ is the performance index to be minimized (e.g., operational cost or carbon emissions); $g_i(\boldsymbol{x};\boldsymbol{\theta})\leqslant 0$ describes the operational boundary conditions, such as equipment output limits and energy storage state-of-charge (SOC) bounds; $h_j(\boldsymbol{x};\boldsymbol{\theta})=0$ corresponds to the physical conservation relationships, such as energy bus power balance equations.

The solution process comprises two stages: (1) a modeling stage that abstracts system information into a mathematical optimization model; (2) a solution stage that translates the model into executable code and invokes solvers to obtain optimal dispatch decisions. Both stages rely heavily on specialized knowledge, and the manual workflow is time-consuming and difficult to scale across diverse engineering scenarios, constraining the widespread adoption of IES optimization technologies.

2.2 MASEO framework

To automate this two-stage process, MASEO is constructed based on three design principles: task decomposition for reducing information overload and ensuring error traceability; guided interaction for mitigating requirement uncertainty and improving information completeness; human-in-the-loop mechanisms for process transparency and reliability. As illustrated in Fig. 3, four sequential specialized agents operate within independent context spaces, exchanging information through structured process files, including a checklist, mathematical model, solution code, and error traceability table. Task type labels (generation or repair) distinguish initial generation from local revision. The agents collaboratively complete information collection, mathematical modeling, code generation, and execution verification, enabling end-to-end automation from natural language input to scheduling optimization results. Specifically, the problem formulation agent (PFA) collects system information through multi-round human–AI dialog and outputs a structured checklist; the energy modeling agent (EMA) constructs the mathematical optimization model based on the checklist; the code generation agent (CGA) transforms the model into executable Python optimization code; the execution and feedback agent (EFA) executes the code, validates physical feasibility, and triggers targeted repair upon failure. Human–AI interaction is concentrated in the PFA stage; subsequent agents proceed autonomously in sequence. The operational workflows are illustrated in Fig. 4. The specific procedures are as follows:

(1) The user submits a dispatch requirement (Step 1a). The PFA interprets user intent and dynamically updates the checklist within the dialog history (Step 1b), then returns the current status, and identifies missing elements (Step 2a), sending status confirmation and targeted inquiries to the user (Step 2b).

(2) The user confirms and supplements information accordingly (Step 3a). Steps 1b–3a iterate until the user confirms that the checklist is complete and accurate. The finalized checklist is transmitted to the EMA (Step 3b).

(3) The EMA retrieves from the historical model repository using the equipment set and bus structure as indices (Steps 4a and 4b), loading matched models as references or returning an empty set. Based on the checklist and retrieval results, the EMA constructs the mathematical optimization model and symbolic mapping table, and then transmits them to the CGA (Step 5a).

(4) The CGA retrieves from the optimization framework knowledge base (Steps 5b and 6a), selects

the most suitable framework, and generates executable Python optimization code, which is transmitted to the EFA (Step 6b).

(5) The EFA executes the code and performs physical feasibility verification (Steps 7a and 7b). If successful, dispatch visualizations are generated, and

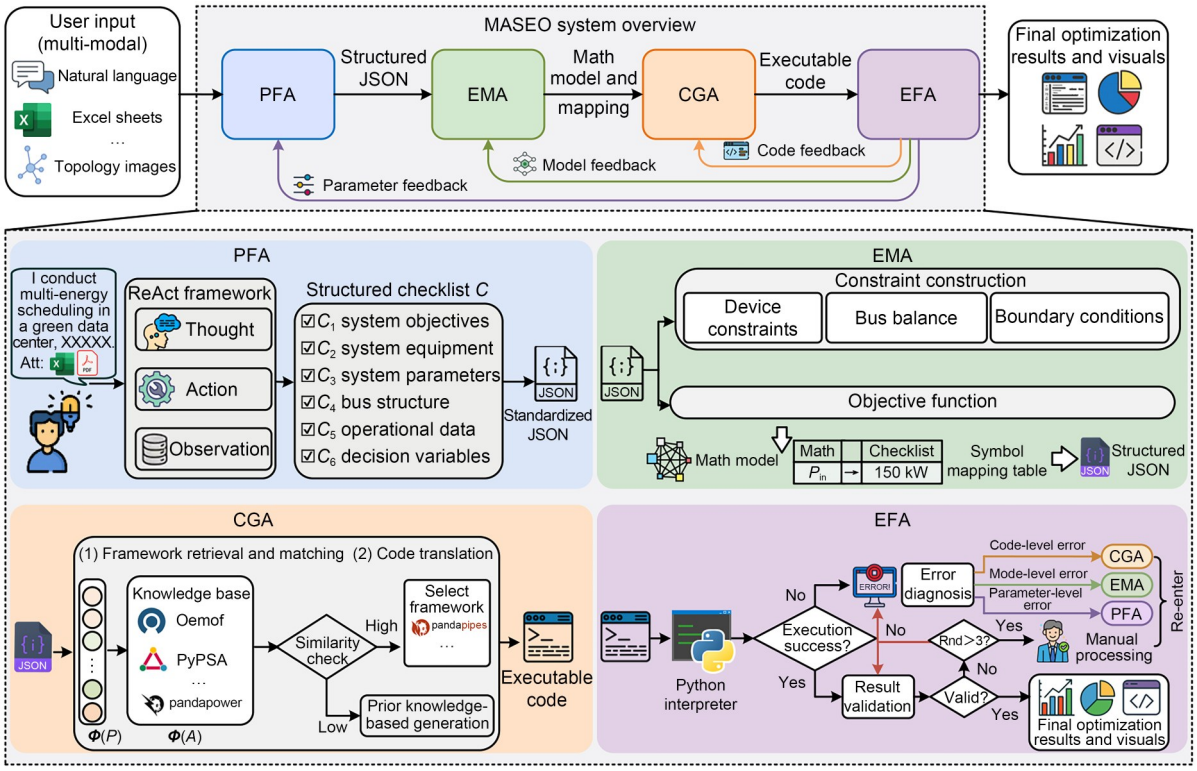


Fig. 3 MASEO system overview. Att is the attachment; JSON is the JavaScript Object Notation; P_{in} is the input power; $\Phi(P)$ and $\Phi(A)$ are the symbolic mappings of P and A, respectively; P and A are the inputs used for similarity calculation in retrieval-augmented generation (RAG); Rnd is the round

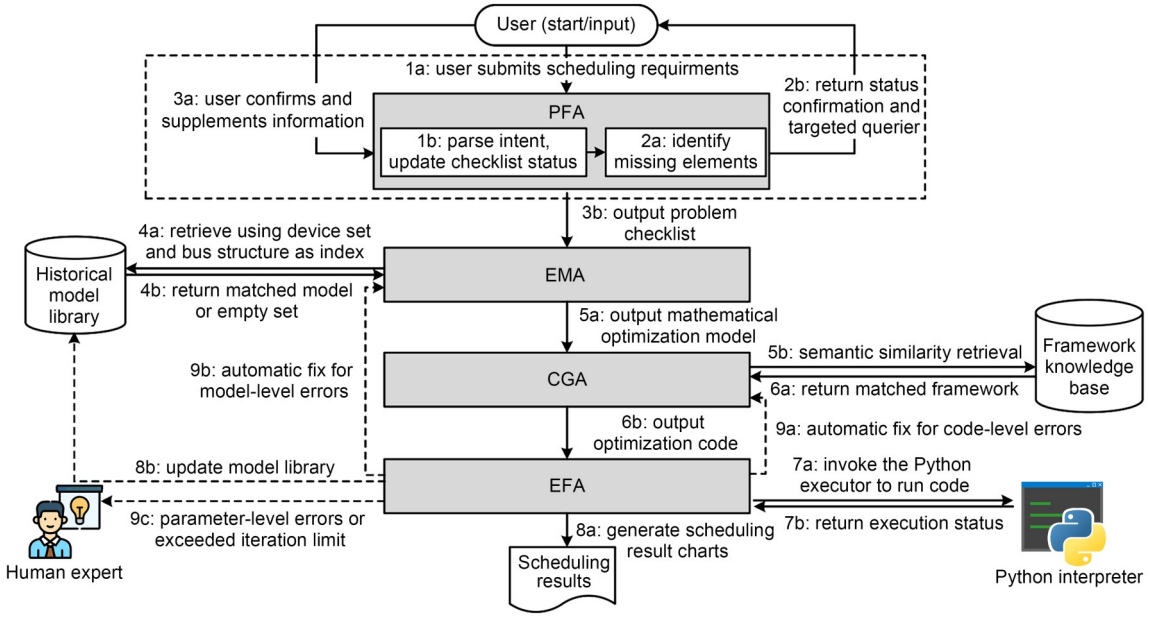


Fig. 4 Workflow diagram of the MASEO framework

the model repository is updated (Steps 8a and 8b). If verification fails or errors occur, an error traceability report is returned to the corresponding upstream agent for localized repair (Steps 9a and 9b). If verification still fails after three iterations or an unrepairable error is encountered, the process terminates, and manual intervention is requested (Step 9c).

2.3 Agent composition of the MASEO framework

Among the four agents, the PFA and EFA require iterative multi-round reasoning and tool invocation, and therefore adopt the ReAct framework; the EMA and CGA perform single-pass reasoning with tool-augmented chain-of-thought, where tools such as retrieval and file operations are invoked as part of the reasoning process but without iterative loops.

2.3.1 Problem formulation agent

The PFA collects complete system information for IES scheduling optimization through multi-round user interaction. Multiple file parsing tools are designed to support information extraction from various file formats, as shown in Section S2.1 of the electronic supplementary materials (ESM). A structured checklist C guides the information collection process, containing six core elements:

$$C=\{C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6\}. \tag{2}$$

Among them, C_5 only stores the path references of time series data files and the statistical features extracted by the parsing tools. The state of C is implicitly maintained in the dialog history H_{PFA}, from which the PFA reconstructs the completion status of each element, identifies missing items, and plans subsequent actions.

As shown in Algorithm S1 in Section S1 of the ESM, the PFA adopts a nested loop structure. The outer loop is triggered by user input, where the PFA reasons over H_{PFA} to identify missing elements and plan actions. If file parsing is needed, the inner loop automatically executes tool invocation and continuous reasoning until a user response is generated. Whenever any element C_i (i=1, 2, ⋯, 6) is updated, the PFA presents results and requests explicit user confirmation. Unconfirmed information is excluded from the valid state. Once all elements are confirmed, the PFA outputs the structured checklist via the file writing tool and transfers it to the EMA.

2.3.2 Energy modeling agent

The EMA takes a structured checklist as input and transforms it into a computationally solvable mathematical optimization model. Unlike the PFA, the EMA addresses inference-intensive tasks with clearly defined and complete input. The tool configuration is shown in Section S2.2 of the ESM.

To reduce redundant reasoning overhead, the EMA first retrieves historical models from the repository using the equipment set and bus structure as indices. If matched, the most structurally similar historical model M^* is loaded into the reasoning context $H_{\mathrm{EMA}}=\{C,\ M^*\}$ for joint model construction; otherwise, $H_{\mathrm{EMA}}=\{C\}$, and reasoning proceeds based solely on the checklist.

The EMA inference comprises two stages. In the mathematical expression generation stage, operational and energy conversion constraints are generated based on device types in C_2; power balance equations are constructed according to the bus structure in C_4; the objective function is formulated based on the optimization objectives in C_1. In the symbol mapping stage, the correspondence between checklist parameters and model symbols is established: decision variables in C_6 map to optimization variables, device parameters in C_3 to known scalars, and time-series data in C_5 to time-varying parameters referenced via file paths and column names. The resulting optimization model is output and passed to the CGA.

2.3.3 Code generation agent

The CGA transforms the mathematical optimization model from the EMA into executable Python solving code, essentially converting decision variables, objective functions, and constraints into code expressions while selecting an appropriate solution method. The tool configuration is shown in Section S2.3 of the ESM.

To mitigate issues such as improper framework selection and incorrect interface calls, a structured knowledge base is constructed from Python-oriented optimization frameworks (e.g., Pyomo, Oemof, PyPSA, and Calliope) to support RAG. Solver invocation rules (e.g., GLPK and Gurobi) are incorporated into the system prompt to constrain code generation.

Specifically, the CGA represents the coding semantics of the current problem as a vector $\phi(P)$ and computes the cosine similarity with the semantic vectors $\phi(A)$ of candidate frameworks in the knowledge base:

$$S(P,\ A)=\frac{\phi(P)\cdot\phi(A)}{|\phi(P)||\phi(A)|}, \tag{3}$$

$$A^{*}=\arg\max_{A} S(P,\ A), \tag{4}$$

where $S(P, A)$ is the similarity score between P and A; A^{*} is the most relevant item retrieved from the candidate set.

CGA selects the framework with the highest score as the reference framework for code generation. When the highest score is lower than the preset threshold τ, the current problem is considered to have insufficient correspondence with existing framework knowledge. In this case, the code generation process does not incorporate retrieval results, and instead constructs the solving code solely based on the mathematical optimization model.

Finally, CGA invokes the file-writing tool to output the generated code and passes it to EFA for execution and validation.

2.3.4 Execution and feedback agent

The EFA executes optimization code and performs state assessment, error diagnosis, feedback-based repair, or result output (Madaan et al., 2023; Diallo et al., 2025). The tool configuration and reasoning workflow are provided in Section S2.4 of the ESM. To ensure traceable error repair, five error categories and their corresponding routing strategies are defined, as detailed in Section S1 of the ESM. EFA executes optimization code through the code execution tool within the user's preconfigured local Python environment; the LLM does not have permission to install dependencies autonomously.

Upon successful execution, the EFA validates physical feasibility by examining device output boundaries and energy conservation constraints. If passed, visualization charts are generated, and process artifacts are archived, with the model metadata written to the historical repository for retrieval and reuse. If execution fails or validation is not satisfied, the error type, message, and relevant file references are recorded in an error tracing table. The task is routed to the corresponding upstream agent for targeted local revision, with modification summaries recorded to enable downstream synchronization, avoiding full regeneration overhead.

The maximum number of automatic repair iterations is 3. If validation still fails or environment-level/information-deficiency errors occur, the EFA archives all process artifacts, notifies the user, and terminates the workflow for manual intervention.

3 Case study

To evaluate the practical effectiveness of the MASEO framework in the scheduling optimization of IES, this study selects the DCIES in Beijing and the CIES in Germany as validation cases.

3.1 Case 1: data center integrated energy system

The Beijing DCIES features hydrogen energy utilization and cascade recovery of data center waste heat, forming a low-carbon multi-energy supply system (Fig. 5). The data center has a rated information technology (IT) capacity of 20 MW (approximately 95% utilization; approximately 19 MW actual load), providing electricity, district heating, and industrial steam services. The supply side integrates photovoltaic, wind power, and hydrogen fuel cells. Waste heat is recovered through cascade utilization: low-grade waste heat (20–40 °C) is upgraded by a low-temperature heat pump, and further converted to steam by a high-temperature steam heat pump. Cooling is jointly provided by electric chillers, absorption chillers, cooling towers, and cold energy recovered from liquid hydrogen regeneration. Thermal and cold storage units smooth load fluctuations.

The operational dataset comprises hourly data for 2022 (8760 steps), covering electricity, cooling, heating, and steam demands, along with meteorological variables. A typical day clustering method extracted 10 representative days. Detailed equipment parameters, representative-day profiles, clustering settings, and time-of-use electricity tariffs are provided in Section S3.1 of the ESM. The carbon emission cost is approximately 0.008 USD/kWh (grid factor: 0.5703 kg/kWh CO_2; carbon price: 13.47 USD/t); the liquid hydrogen price is 0.075 USD/kWh; the external heating, steam, and cooling prices are 0.022, 0.056, and 0.069 USD/kWh, respectively.

3.2 Case 2: community integrated energy system

The German CIES is a combined cooling, heating, and power system centered on a gas turbine, serving the electricity, heating, and cooling demands of an industrial community (Fig. 6). The supply side

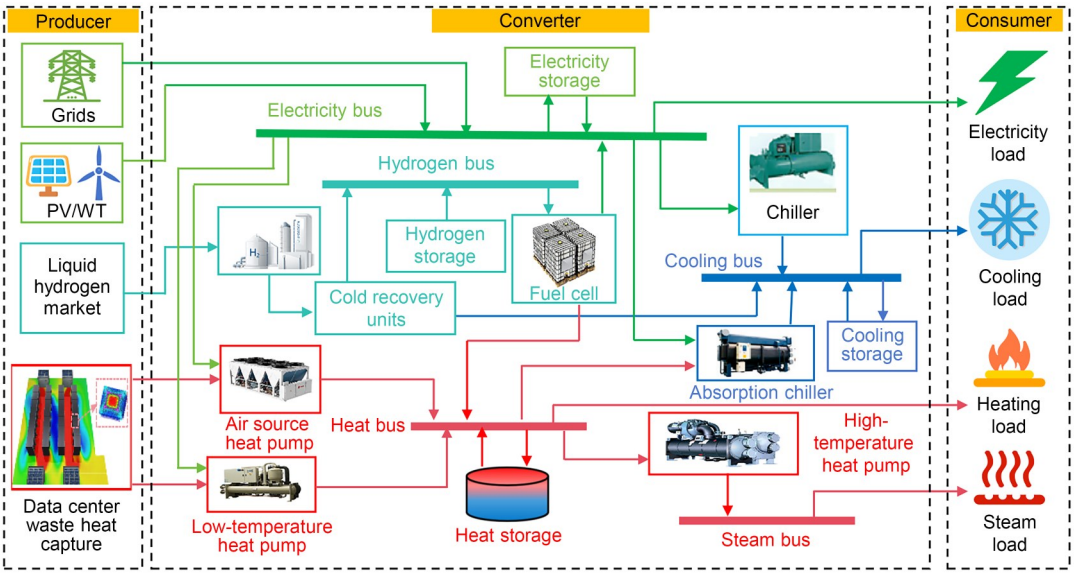


Fig. 5 Energy flow diagram of the Beijing DCIES (Wang SJ et al., 2025). PV is the photovoltaic; WT is the wind turbine

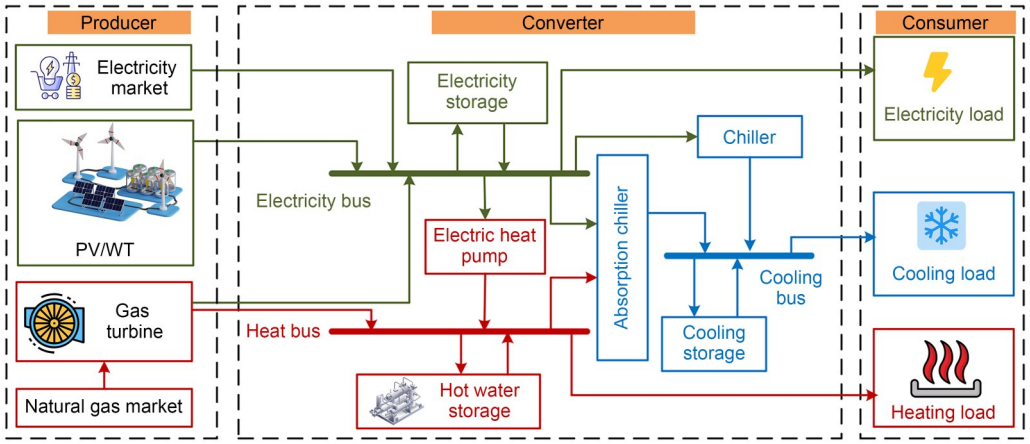


Fig. 6 Structural diagram of the German CIES

integrates photovoltaic and wind power, with a gas turbine as the core cogeneration unit. Waste heat is recovered by an absorption chiller for cooling. An electric heat pump and electric chiller supplement heating and cooling, respectively. Electric, heat, and cooling storage enable peak shaving and load shifting, with grid connections ensuring energy balance.

Operational data consist of hourly time series for the full year, with meteorological data from National Aeronautics and Space Administration (NASA) MERRA-2 (Gelaro et al., 2017) and load data from the Scientific Data dataset (Priesmann et al., 2021). Ten typical days were extracted by clustering. Detailed equipment parameters, representative-day profiles, clustering settings, and electricity pricing are provided in Section S3.2 of the ESM. Electricity pricing follows a two-part tariff: capacity charge 114.29 EUR/(kW·a) and energy charge 0.0025 EUR/kWh. The natural gas price is 0.0286 EUR/kWh.

3.3 Evaluation metrics

The framework is assessed across four dimensions. Expert-annotated references serve as benchmarks, with an LLM-as-Judge approach for scoring to avoid misclassification by rule-based methods. The evaluation results are reviewed and confirmed by domain experts.

3.3.1 Optimization problem modeling accuracy

The capability to extract information and construct mathematical formulations is measured by Eq. (5):

$$S_{\text{modeling}} = \frac{1}{7}\sum_{i=1}^{7} s_i , \tag{5}$$

where S_{modeling} is the overall modeling score; $s_i \in [0, 1]$ (i= 1, 2, ⋯, 7) is the score of the ith evaluation criterion.

Information collection completeness includes equipment completeness s_1, topology accuracy s_2, completeness of parameters and variables s_3, and objective function accuracy s_4. Model completeness includes constraint completeness s_5, accuracy of model representation s_6, and accuracy of parameter mapping s_7. Scores are interpreted in three intervals: [0.60, 0.70] indicates system-level structural errors such as missing bus connections or mismatched equipment; [0.70, 0.85] indicates correct topology but missing local constraints or parameter mapping errors; [0.85, 1.00] indicates high accuracy with only minor differences.

3.3.2 Code execution pass rate

The code execution pass rate is used to reflect the capability of the framework to generate accurate and executable code. The following metrics are defined:

$$R_1 = \frac{N_{\text{success}}^{(1)}}{N_{\text{total}}}, \tag{6}$$

$$R_r = \frac{N_{\text{success}}^{(\leqslant 3)}}{N_{\text{total}}}, \tag{7}$$

where R_1 is the first-pass execution rate (FER); R_r is the post-repair execution rate (PER); $N_{\text{success}}^{(1)}$ is the number of cases successfully executed on the first attempt; $N_{\text{success}}^{(\leqslant 3)}$ is the number of cases successfully executed within three repair attempts; N_{total} is the total number of test cases.

3.3.3 Solution accuracy

Both cases are linear programming problems with guaranteed global optima. Using the expert model solution y^* as a benchmark:

$$\delta = \left| \frac{|y - y^*|}{|y^*|} \right|, \tag{8}$$

where δ is the solution deviation, and the smaller the value, the higher the consistency; y is the framework's objective value.

Framework economic efficiency is evaluated by token consumption and cost per task.

3.4 Case validation

To evaluate usability for low-barrier users, a first-year energy engineering graduate student without IES scheduling optimization experience independently completed both case tasks using only system documentation and the interactive interface, without access to reference model implementation details. The equipment parameters were consistent with those in Sections S3.1 and S3.2 of the ESM. Claude-Sonnet-4.5 was used as the base model (Table 1).

Table 1 Claude-Sonnet-4.5 configuration for case validation

Parameter	Description
Context window	2×10^5 tokens
Max output	16384 tokens
Temperature	0.3
Top-p	0.95

Temperature is used to control the randomness in model generation; Top-p is used to control the range of vocabulary selection during model generation, and it dynamically adjusts the size of the candidate word set based on the cumulative probability distribution

3.5 Framework performance evaluation

An experimental dataset of 13 scheduling optimization task instances was constructed from multiple equipment capacity configurations derived from the planning results of both cases (Section S3 of the ESM). LLMs simulated realistic user input scenarios with partial information provision capability. Unless otherwise specified, all experiments adopted a unified model configuration and default parameters.

To evaluate the overall performance and backbone adaptability, comparative experiments were conducted on several mainstream models under unified hyperparameters (Table 2). A single-agent ReAct baseline using Claude-Sonnet-4.5 with a single context window

Table 2 Pricing information for backbone models

Model	Input price per 10^6 tokens (USD)	Output price per 10^6 tokens (USD)
Llama-4-Scout-17B	0.60	2.40
Qwen3-32B	1.20	5.15
DeepSeek-R1	1.40	5.60
DeepSeek-V3	0.80	3.20
Qwen3-235B-A22B	1.95	8.37
Qwen2.5-72B-Instruct	2.50	7.50
GPT-4.1	2.00	8.00
Claude-Sonnet-4.5	3.00	15.00

completing the entire pipeline was constructed for comparison. Each configuration was run five times with averaged results.

4 Experimental results

4.1 Framework performance evaluation

Application programming interface (API) pricing followed the Claude-Sonnet-4.5 rate (Table 2). As this study does not involve planning, the annual investment cost was set to be identical to the expert model.

4.1.1 Case 1: data center integrated energy system in Beijing

Table 3 compares key economic indicators between MASEO and the expert model. The annualized total cost deviation is approximately 1.4% (13.83×10^6 USD vs. 14.38×10^6 USD). The grid electricity procurement cost decreased by 13.2%, the hydrogen purchase cost increased by 17.1%, and the waste heat recovery revenue increased by 10.4%. These differences stem from reasonable modeling variations in cooling tower treatment: MASEO models it as an adjustable conversion device, granting greater scheduling flexibility. Consequently, the framework tends to reduce grid purchases and increase hydrogen consumption to lower carbon tax expenditures. Such differences fall within a reasonable strategy selection space, and experts can review or adjust cooling tower strategies according to actual scenarios.

The completion time of the end-to-end task was 2.5 h, during which two errors occurred, both automatically corrected. The first, a code-level error caused by a mismatch between Excel column names and code variable names, was routed by the EFA to the CGA. Second, a model-level error from a missing constraint, rendering the cooling tower unavailable above 13 °C, was fed back to the EMA for constraint supplementation. The single-task cost was 1.334 USD. Interaction statistics and token consumption are presented in Table 4.

Fig. 7 presents the energy dispatch results for four typical days, with all energy carriers satisfying the bus balance constraints at every time step. The electricity load remains stable at 18–19 MW, with the hydrogen fuel cell providing a baseload (average 4.79 MW) and renewable output fluctuating seasonally (wind turbine peaks at 15.08 MW in summer; PV peaks at 10.47 MW in transitional seasons). Cooling (average 20.30 MW) exhibits seasonal switching: the cooling tower dominates in winter (15.71 MW), while the electric chiller takes over in summer (17.40 MW). Heating peaks at 14.04 MW in winter and drops to 0.42 MW in summer, primarily from the low-temperature heat pump. Steam peaks at 12.53 MW in winter, mainly from the high-temperature steam heat pump.

4.1.2 Case 2: integrated community energy system in Germany

Table 5 presents MASEO's results for the German CIES. Due to the relatively simple system structure, MASEO generated executable code in a single attempt and passed physical feasibility checks. The annualized total cost is 98.58×10^4 USD, consistent with the expert model. Capacity costs account for the largest share (38.3%), aligning with the two-part tariff commonly applied in German industrial pricing. The single-task API cost was 0.997 USD, lower than Case 1. The end-to-end completion time was approximately 1.5 h. The interaction statistics and token consumption are shown in Table 6.

Fig. 8 illustrates the energy dispatch results for four typical days, with all carriers satisfying the bus balance constraints. The electricity load averages 2033 kW (peak 2678 kW) with a weekday–weekend pattern, primarily grid-supplied. Wind turbine contributes most in transitional winter (peak at 371.4 kW; renewable penetration 14.2%); PV averages 115.5 kW in summer and 26.7 kW in winter. The gas turbine operates only on extreme cold days (−0.6 °C; average 61.1 kW). Heating shows strong seasonality (winter 852.3 kW; summer 577.7 kW), with the heat pump as the core

Table 3 Performance comparison between MASEO and the expert model in the Beijing DCIES

Model	Annualized cost ($\times10^4$ USD)				Surrounding heating cost ($\times10^4$ USD)		Data center waste heat revenue ($\times10^4$ USD)	Total cost ($\times10^6$ USD)
	Capital cost	Grid	Hydrogen	Carbon tax	Heating	Steam		
Expert model	718.00	338.00	460.00	85.00	72.00	167.00	−402.00	14.38
MASEO	718.00	293.35	538.53	83.63	72.81	121.36	−443.91	13.83

Table 4 Interaction statistics and token consumption of the MASEO framework

Metric	Description
Number of total dialog turns	19
Number of PFA dialog turns	11
Number of EMA iterations	2
Number of CGA iterations	3
Number of EFA executions	3
Error encountered	Code-level error; model-level error
Number of input tokens ($\times 10^6$)	0.2550
Number of output tokens ($\times 10^6$)	0.0378
Cost (USD)	1.334

unit. Cooling (average 269.8 kW) features electric chiller (EC)–absorption chiller (AC) complementary operation: EC dominates in winter (91.0%) and transitional winter (65.0%); AC and EC operate jointly in summer (AC 56.0%; EC 44.0%); AC-led in transitional summer.

Case studies demonstrate that the MASEO framework compresses the scheduling optimization work originally performed by domain experts from days or weeks to within hours while maintaining result accuracy consistent with expert benchmarks. The automatic correction of two errors in the Beijing DCIES further validates the effectiveness of the traceable repair

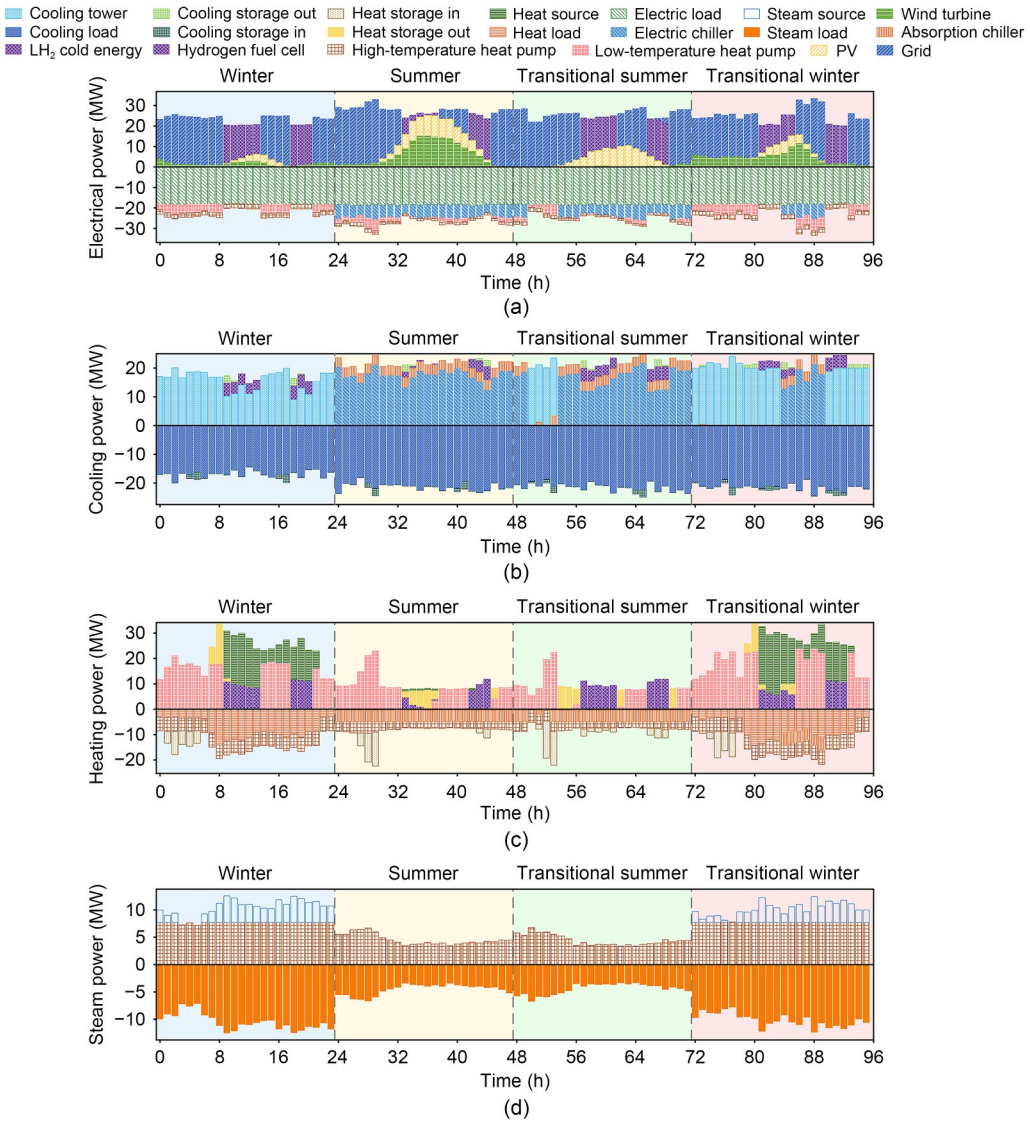


Fig. 7 Hourly energy balance schedules for four representative typical days: (a) electrical balance; (b) cooling balance; (c) heating balance; (d) steam balance. Liquid hydrogen (LH_2) cold energy exists only in the cooling balance, and hydrogen fuel cell exists only in electrical balance and heating balance. Positive values indicate energy supply, and negative values indicate demand

mechanism, partially alleviating the low success rate and heavy manual debugging burden that characterize existing LLM-assisted approaches.

Table 5 Performance of the MASEO in the German CIES

Cost		Value
Annualized cost ($\times10^4$ USD)	Capital cost	56.18
	Grid	4.17
	Gas	0.51
	Capacity	37.72
Total cost ($\times10^4$ USD)		98.58

Table 6 Interaction statistics and token consumption data

Metric	Description
Number of total dialog turns	12
Number of PFA dialog turns	9
Number of EMA iterations	1
Number of CGA iterations	1
Number of EFA executions	1
Error encountered	None
Number of input tokens ($\times10^6$)	0.2010
Number of output tokens ($\times10^6$)	0.0263
Cost (USD)	0.997

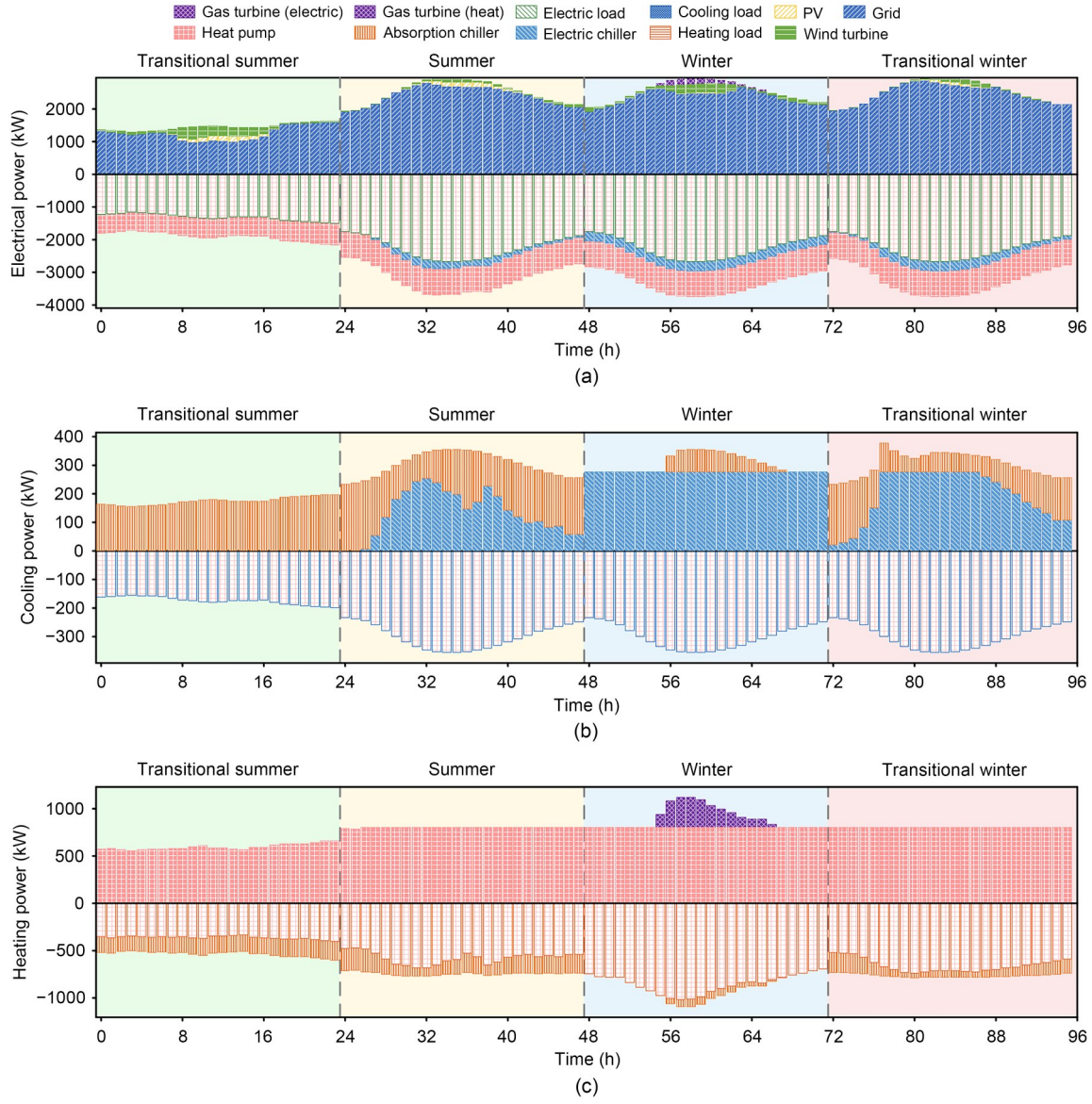


Fig. 8 Hourly energy balance schedules for four representative typical days: (a) electrical balance; (b) cooling balance; (c) heating balance. Positive values indicate energy supply, and negative values indicate demand

4.2 Comparative experiment results

4.2.1 Baseline and ablation analysis

As shown in Fig. 9, the single-agent LLM+ReAct baseline yields failure rates of 40%/37% and S_{modeling} of 0.652/0.668 across the two scenarios. The low accuracy reflects the difficulty of sustaining complete constraint representations over extended dialogs—critical parameters are progressively overwritten, and the resulting omissions propagate into modeling errors. The high failure rates indicate continued reliance on manual debugging, while the large variance reveals that unstructured requirement collection introduces substantial run-to-run inconsistency. MASEO improves S_{modeling} to 0.881/0.897, reduces failure rates to 7%/6%, and narrows solution deviation from 5.84%/5.21% to 1.23%/0.95%, suggesting that the multi-agent framework offers meaningful improvements for complex IES scheduling tasks.

The ablation results reveal the distinct role of each component. −CL causes the largest drop in S_{modeling} (0.748/0.761) and widens solution deviation to 3.21%/2.87% with increased variance, while execution pass rates (FER+PER) remain comparatively stable—confirming that the checklist primarily governs modeling quality and stability. −RP leaves S_{modeling} largely unchanged (0.874/0.889) but raises failure rates to 27%/26% and drops PER from 33%/34% to 17%/17% without error routing. The EFA regenerates blindly, and the same failure patterns recur. −RG mainly reduces FER from 60%/60% to 43%/43% with minimal effect on S_{modeling}. The three modules thus address modeling stability, execution reliability, and code quality as independent dimensions.

Table 7 reports token consumption and per-task cost. FM costs approximately 40.59 USD (Beijing) and 28.62 USD (German), with the difference driven by system complexity and repair iterations. Despite using a single context, the baseline accumulates comparable input tokens due to a long dialog history. Ablation variants incur slightly lower costs from reduced module inputs. Overall, the multi-agent overhead remains modest and economically feasible for deployment.

4.2.2 Backbone model comparison

As shown in Fig. 10, MASEO maintains code execution pass rates above 80% and S_{modeling} above 0.760 across all tested backbone models, demonstrating the

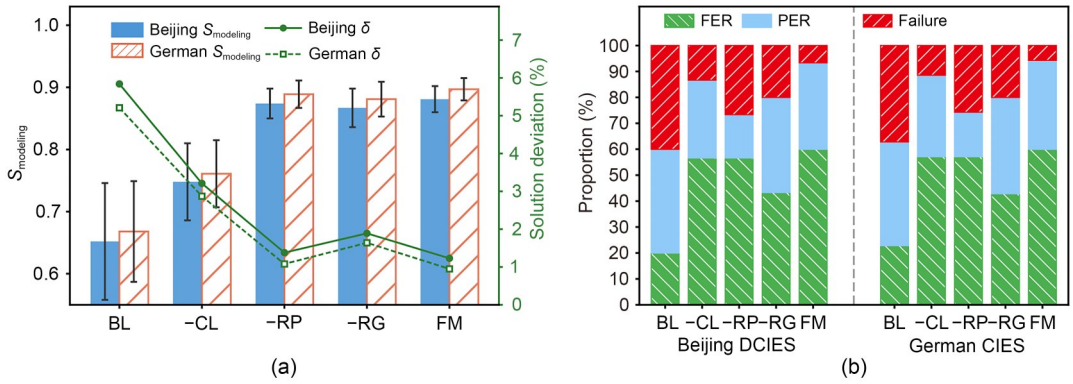


Fig. 9 Baseline and ablation comparison of MASEO across two scenarios: (a) modeling accuracy and solution deviation; (b) execution outcome distribution. BL, −CL, −RP, −RG, and FM represent the single LLM+ReAct baseline, removing the structured checklist, removing the repair mechanism, removing RAG, and full MASEO, respectively

Table 7 Token consumption and per-task cost across different configurations

Configuration	Beijing DCIES			German CIES		
	Input token (×10^6)	Output token (×10^6)	Cost (USD)	Input token (×10^6)	Output token (×10^6)	Cost (USD)
BL	9.82	2.14	42.15	7.24	1.86	32.48
−CL	7.21	1.08	37.14	4.11	0.88	24.03
−RP	7.52	1.02	37.85	4.44	0.85	25.37
−RG	7.84	1.09	38.54	4.28	0.91	25.88
FM	8.03	1.10	40.59	4.96	0.92	28.62

framework's adaptability to different model configurations and providing a reference for model selection in practical deployment. Token consumption and per-task costs are reported in Table 8. Among closed-source models, Claude Sonnet-4.5 (M1) achieves the best overall performance (S_{modeling} 0.881 in Beijing DCIES; failure rate 7%), although at a relatively high per-task cost (40.59 USD/28.62 USD); GPT-4.1 (M2) delivers comparable results at 26.44 USD/18.39 USD, offering a more favorable cost-performance ratio. Open-source large-parameter models (M3–M6) maintain stable performance with S_{modeling} exceeding 0.820 and failure rates of 10%–11%, remaining within an acceptable range relative to closed-source models; among these, DeepSeek-V3 (M4) incurs the lowest cost (21.79 USD/13.56 USD) while sustaining S_{modeling} of 0.826/0.836, demonstrating strong economic feasibility. The lightweight Qwen3-32B (M7) achieves an accuracy comparable to larger open-source models at 23.36 USD/16.60 USD, making it suitable for resource-constrained deployments. Llama-4-Scout-17B (M8) performs relatively poorly, with S_{modeling} dropping to 0.758/0.764 and failure rates rising to 23%/23%, suggesting that the framework imposes a minimum threshold on backbone model reasoning capability. Overall, Claude Sonnet-4.5 or GPT-4.1 is recommended when modeling accuracy is the primary concern; Qwen3-32B or DeepSeek-V3 offers a viable balance of performance and cost under resource constraints; large open-source models are appropriate where both performance and deployment autonomy are needed.

4.3 Sensitivity analysis experimental results

As shown in Fig. 11 and Table 9, the code execution pass rates exceed 80% across all temperature settings T in both scenarios, indicating that MASEO's execution reliability is robust to temperature variation and reflecting the stabilizing role of the traceable repair

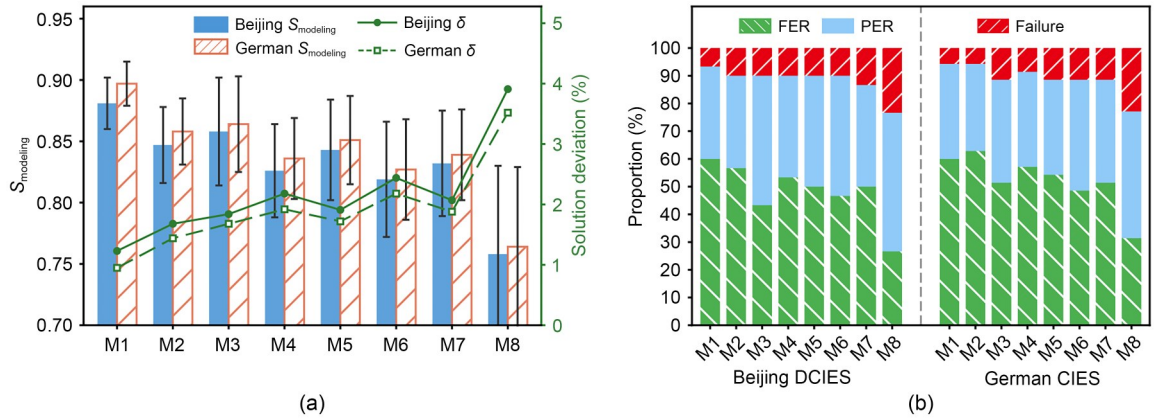


Fig. 10 Backbone model comparison under MASEO: (a) modeling accuracy and solution deviation; (b) execution outcome distribution. M1–M8 denote Claude Sonnet-4.5, GPT-4.1, DeepSeek-R1, DeepSeek-V3, Qwen3-235B-A22B, Qwen2.5-72B-Instruct, Qwen3-32B, and Llama-4-Scout-17B, respectively

Table 8 Token consumption and per-task cost across backbone model configurations

Model	Beijing DCIES			German CIES		
	Input token ($\times 10^6$)	Output token ($\times 10^6$)	Cost (USD)	Input token ($\times 10^6$)	Output token ($\times 10^6$)	Cost (USD)
Claude Sonnet-4.5	8.03	1.10	40.59	4.96	0.92	28.62
GPT-4.1	5.42	0.98	26.44	3.82	0.71	18.39
DeepSeek-R1	6.11	1.41	27.57	5.12	1.24	22.41
DeepSeek-V3	5.11	1.01	21.79	3.48	0.78	13.56
Qwen3-235B-A22B	8.18	1.34	34.78	3.14	0.96	21.32
Qwen2.5-72B-Instruct	6.07	1.24	32.16	4.28	1.02	25.46
Qwen3-32B	4.99	1.05	23.36	3.28	0.84	16.60
Llama-4-Scout-17B	5.14	1.15	26.00	3.42	0.88	13.19

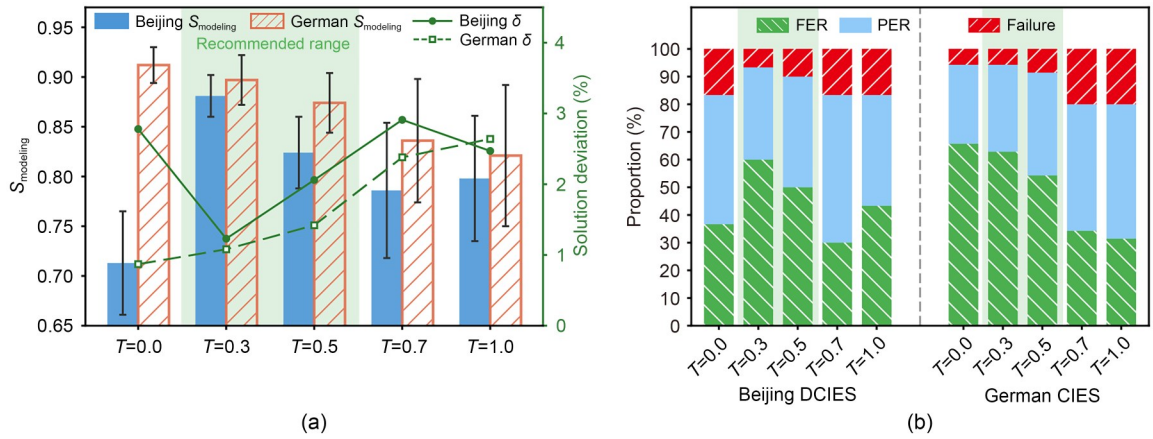


Fig. 11 Sensitivity of MASEO performance at different temperature settings: (a) modeling accuracy and solution deviation; (b) execution outcome distribution

Table 9 Token consumption and per-task cost across different temperature settings

T	Beijing DCIES			German CIES		
	Input token ($\times10^6$)	Output token ($\times10^6$)	Cost (USD)	Input token ($\times10^6$)	Output token ($\times10^6$)	Cost (USD)
0.0	8.74	0.91	39.85	5.61	0.90	30.26
0.3	8.03	1.10	40.59	4.96	0.92	28.62
0.5	7.05	0.99	36.00	4.38	0.88	26.25
0.7	5.20	0.96	30.07	3.77	0.88	24.52
1.0	5.15	1.00	30.49	3.97	0.92	25.66

mechanism. In contrast, temperature has a more pronounced effect on modeling quality, with $S_{modeling}$ ranging from 0.710 to 0.910. Within the lower portion of this range, the model generally preserves the correct system topology but may exhibit local constraint omissions, parameter mapping deviations, or incomplete coupling representations, driving solution deviation above 2.00%. Temperature therefore primarily affects the stability of the requirement parsing and modeling stages rather than the reliability of code execution.

Examining the temperature ranges, T=0.0 produces a clear divergence: the German CIES achieves its highest $S_{modeling}$ (0.912), lowest solution deviation (0.87%), and narrowest error bands; the Beijing DCIES, by contrast, records its lowest $S_{modeling}$ (0.713) and widest deviation (2.78%). This divergence is rooted in the difference in system complexity: an overly conservative generation strategy impairs information completeness in complex scenarios and incurs the highest token consumption (8.74×10^6 tokens for Beijing DCIES), yielding poor economic efficiency. In the range $T\in[0.3, 0.5]$, both scenarios achieve good accuracy, and the stability of the Beijing DCIES reaches an $S_{modeling}$ of 0.881 with a failure rate of 7% at T=0.3, while the German CIES maintains $S_{modeling}$ at 0.897 with token consumption at a reasonable level (7.05×10^6–8.03×10^6 tokens for Beijing DCIES), making this range a favorable balance between performance and cost. As T rises to 0.7 and above, error bands widen, failure rates climb to 17%–20%, and both modeling accuracy and solution deviation deteriorate; high-temperature sampling causes the PFA to terminate information confirmation prematurely, increasing constraint omissions. The reduction in token consumption at high temperatures (approximately 5.2×10^6 tokens for Beijing DCIES) stems from shorter dialogs rather than improved quality, offering limited practical benefit. Overall, $T\in[0.3, 0.5]$ is the recommended working range.

5 Conclusions

This study proposes the MASEO, a multiagent collaborative framework for addressing incomplete requirement elicitation, physical constraint omissions, and error recovery in LLM-assisted IES scheduling

optimization. MASEO decomposes the workflow into four sequential stages supported by a structured checklist and traceable repair mechanism. Across two heterogeneous case studies, MASEO achieves results highly consistent with expert benchmarks, with an annualized cost deviation of approximately 1.4% in the Beijing case. Compared with a single-agent LLM+ReAct baseline, it improves modeling accuracy by approximately 35%, increases the code execution pass rate from 63% to 94%, and reduces solution deviation from 5.84% to 1.23%. These results indicate that workflow-level structural mechanisms are critical to reliable LLM-assisted engineering optimization.

Ablation experiments confirm the complementary contributions of the checklist, repair mechanism, and RAG retrieval to modeling stability, execution reliability, and code quality. Multi-backbone and sensitivity experiments further demonstrate stable performance across different LLM configurations, with $T \in [0.3, 0.5]$ providing a suitable working range.

Current limitations include the focus on day-ahead scheduling, omission of network topology constraints, reliance on structured inputs, and validation in only two scenarios. Future work will extend MASEO to broader IES scenarios, planning–operation optimization, network-aware and containerized deployment, as well as hybrid architectures integrating physics-based models with LLM agents.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2023YFE0108600), the National Natural Science Foundation of China (No. 52576234), the Foundation of Key Laboratory of Cleaner Intelligent Control on Coal & Electricity, Ministry of Education (No. CICCE202510), China, and the Scientific Research Fund of Zhejiang University (No. XY20250460), China.

Author contributions

Boan QU: conceptualization, methodology, software, formal analysis, visualization, and writing–original draft and review & editing. Haoyu JIANG: visualization and writing–original draft. Songjie WANG: conceptualization. Zheng LUO: writing–original draft. Xueru LIN, Xingtao TIAN, Jian LI, Xiaojie LIN, and Wei ZHONG: supervision and writing–review & editing.

Conflict of interest

Boan QU, Haoyu JIANG, Songjie WANG, Zheng LUO, Xueru LIN, Xingtao TIAN, Jian LI, Xiaojie LIN, and Wei ZHONG declare that they have no conflicts of interest.

Declaration on the use of generative AI tools

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

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

di Bella A, Canti F, Prina MG, et al., 2024. Power system investment optimization to identify carbon neutrality scenarios for Italy. *Environmental Research: Energy*, 1(3):035001. https://doi.org/10.1088/2753-3751/ad5b64

Diallo A, Bikakis A, Dickens L, et al., 2025. Rule-guided feedback: enhancing reasoning by enforcing rule adherence in large language models. arXiv:2503.11336. https://doi.org/10.48550/arXiv.2503.11336

Du Y, Wei FY, Zhang HY, 2024. AnyTool: self-reflective, hierarchical agents for large-scale API calls. Proceedings of the 41st International Conference on Machine Learning, article 470.

Fan H, Wang CY, Liu L, et al., 2022. Review of uncertainty modeling for optimal operation of integrated energy system. *Frontiers in Energy Research*, 9:641337. https://doi.org/10.3389/FENRG.2021.641337

Gelaro R, McCarty W, Suárez MJ, et al., 2017. The modern-era retrospective analysis for research and applications, version 2 (MERRA-2). *Journal of Climate*, 30(14):5419-5454. https://doi.org/10.1175/JCLI-D-16-0758.1

Hilpert S, Kaldemeyer C, Krien U, et al., 2018. The open energy modelling framework (OEMOF)-a new approach to facilitate open science in energy system modelling. *Energy Strategy Reviews*, 22:16-25. https://doi.org/10.1016/j.esr.2018.07.001

Huang SJ, Lu H, Chen MZ, et al., 2023. Integrated energy system scheduling considering the correlation of uncertainties. *Energy*, 283:129011. https://doi.org/10.1016/j.energy.2023.129011

Jia MS, Cui ZY, Hug G, 2024. Enabling large language models to perform power system simulations with previously unseen tools: a case of daline. arXiv:2406.17215. https://doi.org/10.48550/arXiv.2406.17215

Jiang HY, Zeng FJ, Qu BA, et al., 2026a. Helios: a foundational language model for smart energy knowledge reasoning and application. Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), p.3208-3220. https://doi.org/10.18653/v1/2026.eacl-long.148

Jiang HY, Qu BA, Zhu JJ, et al., 2026b. HyperLoad: a cross-modality enhanced large language model-based framework for green data center cooling load prediction. Proceedings of the 40th AAAI Conference on Artificial Intelligence, p.480-488. https://doi.org/10.1609/aaai.v40i1.37011

Jiang JY, Wang F, Shen JS, et al., 2026. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 35(2):58. https://doi.org/10.1145/3747588

Jin M, Sel B, Hardeep F, et al., 2023. A human-on-the-loop optimization autoformalism approach for sustainability. arXiv:2308.10380. https://doi.org/10.48550/arXiv.2308.10380

Li R, Allal LB, Zi YT, et al., 2023. StarCoder: may the source be with you! arXiv:2305.06161. https://doi.org/10.48550/arXiv.2305.06161

Lin XJ, Luo Z, Du-Ikonen L, et al., 2025. Generative artificial intelligence: pioneering a new paradigm for research and education in smart energy systems. *Energy and AI*, 22:100610. https://doi.org/10.1016/J.EGYAI.2025.100610

Liu MX, Shi Y, Fang F, 2014. Combined cooling, heating and power systems: a survey. *Renewable and Sustainable Energy Reviews*, 35:1-22. https://doi.org/10.1016/j.rser.2014.03.054

Liu SC, Chen CS, Qu XH, et al., 2024. Large language models as evolutionary optimizers. IEEE Congress on Evolutionary Computation, p.1-8. https://doi.org/10.1109/CEC60901.2024.10611913

Madaan A, Tandon N, Gupta P, et al., 2023. Self-refine: iterative refinement with self-feedback. Proceedings of the 37th International Conference on Neural Information Processing Systems, article 2019.

Majidi H, Hayati MM, Breyer C, et al., 2025. Overview of energy modeling requirements and tools for future smart energy systems. *Renewable and Sustainable Energy Reviews*, 212:115367. https://doi.org/10.1016/j.rser.2025.115367

Michelon F, Zhou YH, Morstyn T, 2025. Large language model interface for home energy management systems. Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems, p.590-602. https://doi.org/10.1145/3679240.3734586

Priesmann J, Nolting L, Kockel C, et al., 2021. Time series of useful energy consumption patterns for energy system modeling. *Scientific Data*, 8(1):148. https://doi.org/10.1038/s41597-021-00907-w

Roziere B, Gehring J, Gloeckle F, et al., 2023. Code Llama: open foundation models for code. arXiv:2308.12950. https://doi.org/10.48550/arXiv.2308.12950

Song DR, Meng WQ, Dong M, et al., 2022. A critical survey of integrated energy system: summaries, methodologies and analysis. *Energy Conversion and Management*, 266:115863. https://doi.org/10.1016/j.enconman.2022.115863

Wang SJ, Xiang D, Zhong W, et al., 2025. A multi-objective bilevel planning for data center integrated energy systems with waste heat utilization. *Energy*, 335:137934. https://doi.org/10.1016/j.energy.2025.137934

Wang YD, Wang JY, Chu ZW, 2025. Multi-agent large language models as evolutionary optimizers for scheduling optimization. *Computers & Industrial Engineering*, 206: 111197. https://doi.org/10.1016/j.cie.2025.111197

Yang CR, Wang XZ, Lu YF, et al., 2024. Large language models as optimizers. arXiv:2309.03409. https://doi.org/10.48550/arXiv.2309.03409

Yang YF, Zhou ZC, Xiao XB, et al., 2024. Intelligent day-ahead optimization scheduling for multi-energy systems. *Frontiers in Energy Research*, 11:1349194. https://doi.org/10.3389/fenrg.2023.1349194

Zeng AH, Xu B, Wang BW, et al., 2024. ChatGLM: a family of large language models from GLM-130B to GLM-4 all tools. arXiv:2406.12793. https://doi.org/10.48550/arXiv.2406.12793

Zhu HH, Wang XL, Wen YT, et al., 2025. A review of integrated energy system modeling and operation. *Applied Energy*, 400:126572. https://doi.org/10.1016/j.apenergy.2025.126572

Electronic supplementary materials

Sections S1–S5, Algorithms S1 and S2, Tables S1–S14, Figs. S1–S8