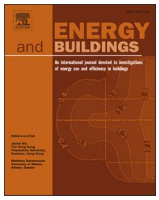
Energy & Buildings 349 (2025) 116516

Contents lists available at ScienceDirect

Energy & Buildings

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



AutoBEE: A hierarchical multi-agent approach for energy and environmental parameter analysis

Yani Quan [a,1], Tong Xiao [a,1], Jiefan Gu [b,1], Peng Xu [a,*,1]

[a] School of Mechanical Engineering, Tongji University, Shanghai 200092, China
[b] College of Architecture and Urban Planning, Tongji University, Shanghai 200092, China

ARTICLE INFO

Keywords:
Large Language Model (LLM)
Multi-Agent System (MAS)
Building energy consumption analysis
Building environmental parameters

ABSTRACT

Traditional approaches to building performance analysis often rely on manual or semi-manual methods, suffering from cumbersome workflows, low efficiency, and high error rates. This study developed AutoBEE, an automated analysis framework for building energy consumption and environmental parameters based on hierarchical multi-agent system integrated with large language models. This framework focused on enhancing the efficiency of individual agent and productive collaboration among agents group. Through the development of a comprehensive agent tool library, the establishment of a multi-level network spanning from teams to agents, the design of a lightweight communication protocol, and the creation of dynamic path planning, AutoBEE achieves autonomous unmanned operation from natural language input to building performance report output. During operation, agents group can exhibit a wide range of capabilities, including but not limited to accurately parsing user instructions, decomposing complex tasks into subtasks, adjusting parameters in the input data file, executing simulation calculations by energy simulation software, and generating structured professional reports. Verified through experiments in 54 typical scenarios, compared with traditional methods, AutoBEE has significantly improved efficiency, accuracy, rationality, content richness, and economic feasibility, providing an innovative solution for building performance research.

1. Introduction

1.1. Research background

Driven by the global energy crisis and the wave of green building, building energy consumption and performance analysis (BECPA) has become core methods to promote sustainable development in the construction sector [1]. BECPA is applied throughout the whole architecture process, covering architectural design optimization, energy efficiency evaluation of mechanical and electrical systems, energy-saving renovation of existing buildings, and building operation performance adjustment. It provides a scientific basis for energy-saving decisions and environmental quality improvement in all links through quantitative data. [2] However, due to the high complexity of tasks, low degree of automation, and heavy reliance on manual operations, BECPA faces challenges of strong professionalism, intricate workflows, and time-consuming tasks in engineering applications.

In engineering applications, the BECPA process can be broadly divided into two parts: firstly, acquiring energy consumption or environmental parameter data; secondly, analyzing the data and presenting it in reports or other forms. For the data acquisition process, current mainstream methods include field measurements, statistical analysis, and computer simulations. Field measurements rely on sensors to collect real time data but face issues with spatial coverage and data continuity [3]. Statistical analysis constructs predictive models based on historical data but struggles to adapt to complex and dynamic real world scenarios [4]. In contrast, Building Energy Modeling (BEM) technology, with its ability to accurately predict energy consumption and environmental

Abbreviations: LLM, Large Language Model; MAS, Multi-Agent System; BECPA, Building energy consumption and performance analysis; BEM, Building Energy Modeling; CSV, Comma-Separated Values; JSON, JavaScript Object Notation; HVAC, Heating, Ventilation and Air Conditioning; BIM, Building Information Modeling; RAG, Retrieval Augmented Generation; IDF, Input Data Files; API, Application Programming Interface; AutoBEE, Automated building energy and environment analysis; PMV, Predicted Mean Vote; COP, Coefficient of Performance; CFD, Computational Fluid Dynamics; PDF, Portable Document Format.

* Corresponding author at: School of Mechanical Engineering, Tongji University, Shanghai 200092, China.
E-mail addresses: quanyanibjut@163.com (Y. Quan), pi620903@163.com (T. Xiao), gu_jiefan@tongji.edu.cn (J. Gu), xupeng@tongji.edu.cn (P. Xu).
[1] Permanent address: Room A434, School of Mechanical Engineering, Tongji University, Jiading District, 200092, Shanghai, China.

https://doi.org/10.1016/j.enbuild.2025.116516
Received 23 May 2025; Received in revised form 15 September 2025; Accepted 23 September 2025
Available online 25 September 2025
0378-7788/

parameters, has provided reliable support for decision making throughout the building lifecycle process. However, BEM technology also faces challenges in its application in multiple dimensions [5]. On the one hand, constructing feasible energy consumption models requires practitioners to have a comprehensive understanding of relevant knowledge, especially, modern buildings integrate complex mechanical systems and new building materials with the innovation of the construction industry, making the modeling process more comprehensive. As shown in the list of energy efficiency simulation elements compiled by Mendes, its rich content vividly illustrates the complexity of building energy modeling and model modification work [6]. On the other hand, there are significant differences in the functionality and operational logic of BEM software available on the market, which leads to poor interoperability between software, requiring users to relearn each time they engage with a new software. Based on the above two points, even professional and skilled BEM engineers often spend enormous time and effort to complete a reliable BEM model. After BEM modeling and simulation are completed, for the data analysis process, data output from BEM models or other channels features large volume, multiple dimensions, and diverse modalities. When tackling specific issues, engineers also need to incorporate additional knowledge inputs like regulations and standards. As a result, engineers spend much effort extracting valid information from the data and synthesizing it into reports. Meanwhile, it is hard to fully ensure the accuracy, rationality, and content richness of the reports. Take the architectural design phase as an example. If an engineer needs to evaluate the impact of envelope design changes on expected energy consumption, the following cumbersome processes are required. First, it takes a lot of time to build a BEM model, and the modeling process is prone to errors due to parameter setting or geometric description deviations. After completing the baseline scheme simulation, it is necessary to manually modify the envelope parameters in the model (such as the thermal performance of wall materials, window-to-wall ratio, etc.) and rerun the simulation. Finally, it is also necessary to filter and extract energy consumption-related data from the multi-format files (such as CSV, JSON, or text reports) output by BEM to complete the comparative analysis of the old and new schemes. In the building operation phase, when engineers need to analyze whether a building complies with certain energy-saving codes, in addition to repeating the modeling and data analysis processes, they also need to invest a lot of energy in retrieving and comparing industry codes, and cross-validating the code requirements with the simulation results. In summary, the BECPA process is extremely cumbersome and highly professional.

Considering the complexity of BECPA modeling and data processing, exploring user-friendly, fully automated, and high quality BECPA methods is core critical to significantly improving work efficiency and the quality of BECPA analysis. It is obvious that automating data collection, model construction, and parameter analysis through intelligent algorithms and programs can significantly improve the efficiency and accuracy of the BECPA process [7]. However, at present, there is no tool or software on the market that can fully automate and cover all these processes with the focus remaining on optimizing the modeling front-end interface, most of the work still needs to be handled manually according to the workflow. Upon detailed analysis, the reason lies in that most traditional algorithms rely on preset rules and can only handle specific scenarios step by step as programmed, while the BECPA process has distinct characteristics. In terms of model input, the construction of physical models and the determination of dynamic boundary conditions are inherently complex. Moreover, as users' focuses vary widely, covering aspects such as building envelopes and operating systems, it is extremely challenging for rule-based programs to fully accommodate the diverse requirements of various building scenarios. Regarding model output, the data does not follow a fixed format but rather exhibits diversity, making it difficult for rule-based programs to extract information that can address users' problems. Additionally, the communication barrier between users' natural language and professional modeling remains a crucial bottleneck that urgently needs to be overcome.

With the advancement of artificial intelligence technology, the emergence of Large Language Model (LLM) and intelligent agents has brought new opportunities for BECPA, particularly to address the challenges posed by traditional algorithms above [8]. LLMs can understand a variety of user needs and convert them into technical instructions [9]. And intelligent agents based on LLMs, through modular design and the integration of components such as memory, planning, and tool invocation, can autonomously decompose and execute tasks, rather than relying on originally preset rule-based steps [10]. Specifically, multi-agent systems can handle more complex tasks through division of labor and collaboration. The combination of LLMs and agents provides a completely new technological pathway for automated building energy efficiency analysis, driving the field into a new stage of intelligent development [51112].

1.2. Literature review

In the advancement of LLMs, scholars have increasingly explored the applications in the field of building energy and environment. LLMs show significant potential in various building performance scenarios, such as intelligent control systems, code generation, and regulatory compliance, as analyzed by Zhang et al. [5] in 2023. However, challenges like high computational costs, data privacy concerns, and fine-tuning complexity impede their application. To tackle these issues, researchers have employed diverse strategies. Many scholars [13–20] have conducted research on different aspects of LLM application, such as exploring LLM's knowledge in the Heating, Ventilation and Air Conditioning (HVAC) industry, enriching information from Building Information Modeling (BIM), HVAC terminal control, and information query system development. The specific details of these directions are summarized in Table 1. Meanwhile, methods such as Retrieval Augmented Generation (RAG) and fine-tuning have been used to enhance LLM capabilities. In summary, these studies demonstrated LLMs' potential in building energy applications [2122].

In the field of BECPA from BEM relevant to this study, the application research with LLMs has emerged as a prominent focus. Regarding the

Table 1
Overview of Existing LLM Research in Building Energy.

LLM Application in building fields	References	Key Research Content
Model Evaluation	[13]	Evaluation of LLM capabilities in the HVAC field.
	[14]	Evaluation of LLM capabilities in building energy retrofit.
Building Retrofit	[11]	Utilization of LLMs for building energy retrofit.
Building Control	[15]	Utilization of LLMs for building control.
	[18]	Utilization of LLMs for interpretable machine learning control.
Building Energy Management	[16]	Utilization of LLMs for building operation information query.
	[17]	Utilization of LLMs for intelligent building management.
BIM Information	[20]	Utilization of LLMs for building BIM information search.
Building Energy Modeling	[5]	Utilization of Multi-agents for building energy consumption simulation.
	[19]	Utilization of LLMs for data-driven urban building energy modeling.
	[23]	Utilization of LLMs for IDF files without geometric information.
	[24]	Utilization of Fine-Tuning LLMs for building energy simulation modeling.
	[25]	Utilization of feedback mechanism by LLMs for optimizing IDF Files.
	[26]	Design of architecture structuring with LLMs to simplify building energy analysis development process.

BEM modeling process, many scholars have explored paths to automate BEM operations, with the automatic generation of Input Data Files (IDF) at the core, complemented by simulation methods. Zhang et al. [23] focused on geometry-free IDF file generation, systematically integrating relevant parameters into LLM prompt templates via analyzing design requirements, extracting standard parameters, and retrieving similar models from HVAC design specifications and the EnergyPlus model library to enable automatic generation. Jiang et al. [24] further customized LLMs through fine-tuning, enabling them to understand users' natural language inputs with simple geometric information and simulation requirements. Additionally, Zhang et al. [525] constructed a feedback mechanism, transmitting the simulation results from the simulation software back to the LLMs to optimize and correct IDF files. This ensured the accurate conversion of building descriptions into error-free EnergyPlus models. These IDF inputs were then processed via the simulation software's API, followed by model simulation and result output. Regarding the data analysis process, Zhang et al. [5] demonstrated the ability of LLMs to plot simulation results in specific dimensions. However, the research findings did not demonstrate that LLMs can extract multi-dimensional data required by users from complex results.

Although the above mentioned studies covered various aspects of simulation input, execution, and post-processing, they still struggled to provide end-to-end solutions when dealing with complex problem inputs from users in multiple scenarios due to the insufficient technical integration and stage coordination. Specifically, most existing research focused on technical breakthroughs in single links of the energy consumption simulation process (such as file generation, simulation execution, or result analysis), but BECPA was inherently a complex process with closely linked multi-link and multi-stage components. Due to the lack of organic integration of technologies across stages, the entire process from user requirement input to final result output could not be smoothly connected and coordinated.

To address the challenge of technical integration, scholars also made corresponding attempts. In these efforts, multi-agent systems were regarded as highly promising solutions. Through division of labor, collaboration, and information exchange, multi-agent systems could organically connect technologies at various stages and dynamically decompose and plan complex tasks. Zhang et al. [525] introduced multiple agents to handle different tasks within the full-process application framework of LLM-based simulation, and also proposed a structured architecture and open-source library to simplify the development, sharing, and deployment of LLM agents for building energy analysis and modeling [26]. Similarly, Xiao et al. [11] utilized agents with different roles to complete the entire workflow of automated building energy optimization, from extracting information from unstructured audit reports and generating building metadata to providing energy efficiency diagnosis and retrofit recommendations. The specific research work of scholars can be found in Table 1. However, these studies failed to fundamentally provide a highly robust complete framework for BECPA. Researchers focused on the performance of each agent within its task scope rather than the collaborative capability of agent groups throughout the entire task process. Specifically, the operation sequence and logic of agents still relied on predefined procedural steps or required manual intervention to operate the agents, rather than autonomously planning execution paths based on tasks to achieve the requirements of unmanned operation and user-friendliness. For LLMs, their proactive task decomposition, planning, and execution mechanisms were the key to outputting reliable reports across multiple scenarios. Meanwhile, according to investigations, the current capabilities of single agents still did not meet the requirements for adapting to multi-scenario and multi-task environments, but were more targeted at single and simple scenarios.

1.3. Research objectives and technical challenges

In order to solve the research questions pointed out in Section 1.2, this study aimed to build an automated, intelligent and user-friendly BECPA analysis framework named AutoBEE (Automated Building Energy and Environment Analysis) based on the multi-agent collaborative system and BEM software. AutoBEE focuses on two core directions: first, the collaborative capability of the agent group achieved through information interaction. By dynamically selecting paths, the entire process operates efficiently and automatically, completely eliminating the need for manual intervention; The second is to strengthen the execution efficiency of a single agent. The core goal of the research is to break through the bottleneck of traditional AI algorithms, which are highly dependent on step-by-step guidance from users. By AutoBEE, Users only need to input natural language instructions about BECPA, without having to master complex building energy consumption expertise or be familiar with professional software, the system can automatically complete the whole chain of work from task decomposition, parameter adjustment, simulation calculation, result analysis and report output. For example, during the architectural design phase, users only need to input a question like "the impact of changing a certain parameter on energy consumption" to obtain the expected energy consumption change results, thereby assisting in decision-making, without the need to learn software or perform data processing.

It is important to emphasize that this paper focuses on the construction and optimization of the automated analysis process. As the generation of IDF files has been addressed by numerous scholars and is not the core of this study, it is assumed that users can provide original IDF files as the analytical basis. Given that EnergyPlus is the most widely used building energy simulation software in the industry (adopted in 65 % of related studies) [6], it was selected as the core simulation tool for this framework to fully leverage its high-precision simulation capabilities.

To achieve the goals of full automation and intelligence in BECPA, the following three key technical problems need to be overcome ((1) and (2) focus on the interaction among agent groups;(3) focus on the effectiveness of individual agents.):

(1) **The problem of agent task allocation**: In the BECPA scenario, tasks that users need to complete often involve complex processes across multiple scenarios and dimensions. For example, to achieve a specific building energy consumption analysis requirement, it may be necessary to sequentially complete core tasks such as IDF file modification, simulation calculations, and data analysis. Among these, IDF file modification is further subdivided into professional subtasks such as building envelope parameter adjustment, HVAC system configuration optimization, and power equipment operation parameter setting. Data analysis covers multi-level tasks including data cleaning, multi-source data integration, multi-dimensional data analysis, standard specification matching, report text compilation, and visualization. More complexly, when tasks involve multi-scenario comparisons (such as energy consumption simulations of two design schemes), key tasks like modification and simulation need to be executed multiple times. In such dynamic task chains, the core challenge for agent groups is: how to autonomously plan dynamic execution paths based on the specific requirements and real-time status of tasks, and accurately match subtasks to agents with corresponding capabilities, ensuring efficient collaboration throughout the process and achieving qualified results output.

(2) **The problem of communication mechanism optimization**: When addressing the above task allocation problem, efficient communication between agents serves as a core prerequisite. However, frequent communication among agents incurs high token costs, and the transmission of invalid information may interfere with task execution. In the BECPA task, each agent generates substantial information when executing each subtask. The information existing in the final system includes details of every IDF file modification, specifics of executed operations, and the entire process of report generation. If such

information is fully transmitted among all agents, it will obscure critical information and affect the agents' accurate judgment of task progress. Therefore, the core issue is transformed into: How to construct an efficient communication mechanism between agents that can both mitigate task execution risks and minimize token consumption in large models, thereby effectively controlling communication costs.

(3) **The problem of improving the execution efficiency of individual intelligent agents:** After tasks are accurately allocated to agents, their execution capabilities directly determine the final completion effect of the tasks. In highly specialized fields such as BECPA, single agents generally face three core challenges when handling complex professional tasks: narrow execution scope, insufficient execution capabilities, and potential "hallucination" risks. For example, in scenarios involving modifications to HVAC system temperature settings in IDF files, agents often cannot directly provide scientifically reasonable modification methods due to limitations in existing knowledge. Therefore, the technical difficulty lies in: how to enhance the execution capabilities of single agents to enable them to achieve precise and efficient processing of complex professional tasks and ensure task success.

In response to the above challenges, this study plans to conduct explorations in the following chapters: Chapter 2 will elaborate on the technical route and key technologies of the AutoBEE framework; Chapter 3 will construct 54 typical scenario experiments to verify the significant advantages of the AutoBEE framework compared with traditional methods; Chapters 4 and 5 will summarize the research results and deeply discuss the research limitations and future development directions. The research results of this study will provide innovative solutions for building performance research, promote the development of BECPA towards the direction of high efficiency, intelligence, and user-friendliness.

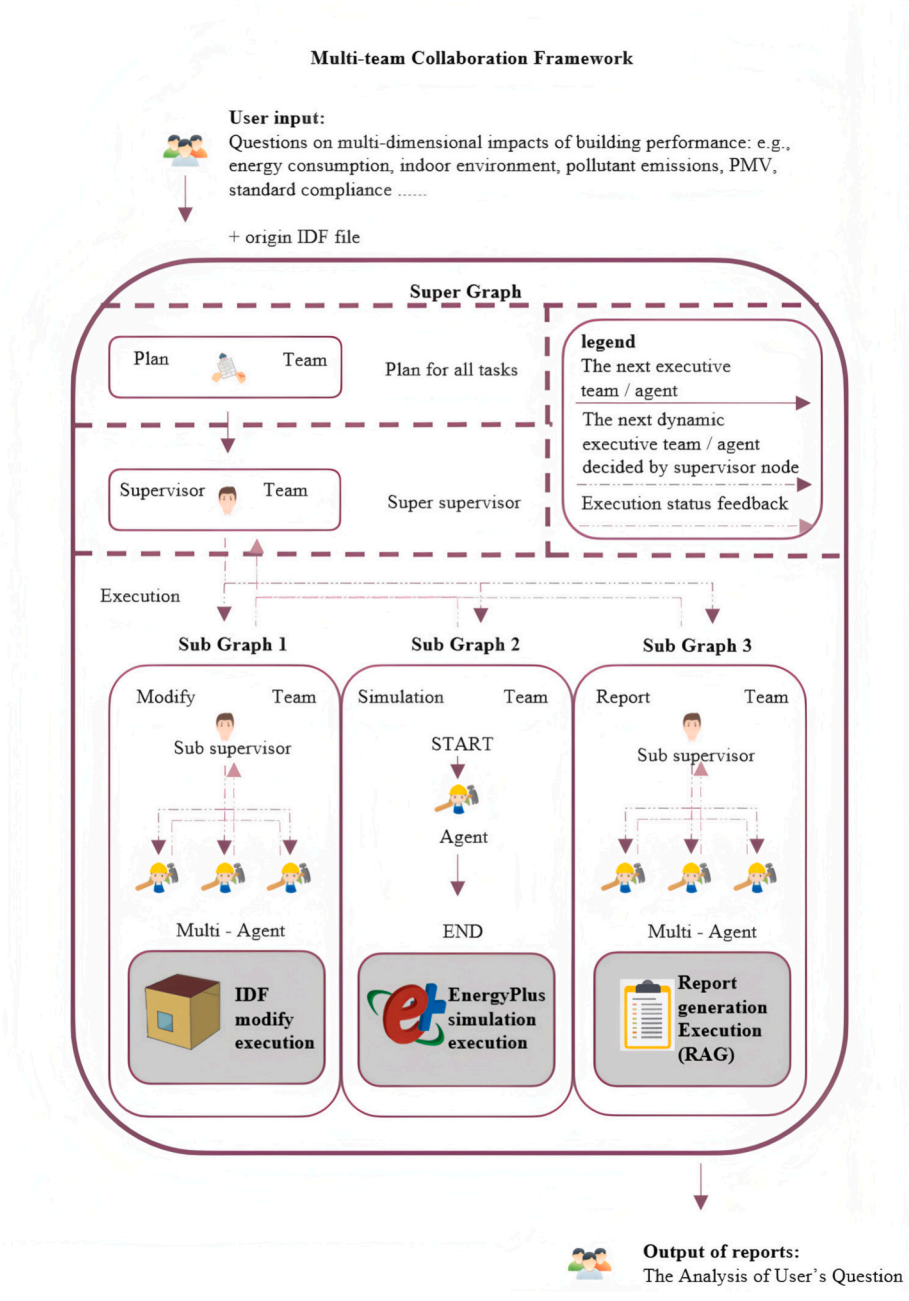


Fig. 1. Multi-agent collaborative framework for building performance analysis.

2. AutoBEE framework

2.1. Technical route

To achieve the goal of fully automating BECPA, this study focuses on multi-agent system leveraging LLM to build an end-to-end system named AutoBEE that specifically addressed three key technical challenges mentioned in Section 1.3: agent task allocation, communication mechanisms, and individual execution efficiency. A series of innovative solutions are proposed:

(1) Enhanced agent execution through the construction of the BECPA tool library and other technologies [271228];

(2) Constructed a multi-level work network composed of teams and agents;

(3) Optimized multi-level super node architecture for team collaboration [29];

(4) Optimized lightweight communication protocol for economy and robustness [30].

The specific technical details will be described in Section 2.2. Fig. 1 is the overall framework of AutoBEE.

As shown in Fig. 1, the overall framework structure of AutoBEE is presented. Table 2 outlines the specific meanings of Super Graph, team, Sub Graph, nodes, edges, super nodes, and path selection under this framework, with its technical principles detailed in Section 2.2. The entire framework is composed of multiple work teams, including a planning team, a supervision team, a modification team, a simulation team, and a reporting team, with each team consisting of its specific agents. This multi-level structure is conducive to the precise allocation of tasks. Within this framework, the user first inputs relevant questions involving multiple dimensions of building performance, including, but not limited to, energy consumption, indoor environmental conditions, pollutant emissions, predicted mean vote (PMV), and compliance, while also providing the original IDF file. At the "Super Graph" level, The "Planning team" is responsible for formulating a global original execution plan according to task characteristics, ensuring the logicality of subsequent task execution processes. The "Supervisor team" oversees the entire BECPA task and dynamically selects the next execution team at the "Sub Graph" level based on the received real-time feedback on execution status and the obtained plan list. During the execution phase, each "Sub Graph" also contains sub-super nodes that dynamically select the next agent to execute. In "Sub Graph 1," the supervisor responsible for modifying tasks leads a multi-agent team to make precise modifications to the parameters of the IDF file according to predefined requirements. In "Sub Graph 2," a specific agent triggers the simulation program based on the IDF file, completing the task when the simulation is finished. In "Sub Graph 3," the supervisor responsible for reporting organizes a multi-agent team to comprehensively collect, analyze, and integrate the simulation data.

The highlights of AutoBEE lie in enabling efficient collaboration among agent groups and enhancing the capabilities of individual agents. In terms of agent groups collaboration, the framework first organizes agents with similar functions into teams to form a multi-level task completion network. Additionally, the framework employs super nodes composed of LLMs at various levels to dynamically plan the next execution team or agent. Meanwhile, a lightweight communication protocol is designed to reduce operational costs and minimize information interference. Regarding the capabilities of individual agents, technologies such as CoT and RAG are significantly utilized, and a complete omnidirectional tool library covering IDF file modification, simulation, and report analysis is built for enhancement.

AutoBEE can achieve fully automated BECPA. For example, if a designer needs to test the energy-saving potential of a new building material, they only need to input the material properties and problem description. AutoBEE will then automatically execute the aforementioned process and provide a energy consumption comparison between the new and old materials, assisting the designer in making decisions. The specific technical implementation details mentioned above are detailed in following Section 2.2.

2.2. Analysis and explanation of key technologies

The following sections provide a detailed elaboration on these key technical methodologies.

2.2.1. Agent execution capability

To enhance the execution capabilities of agents, this study establishes a complete library of building performance analysis tools while integrating the CoT technique and RAG technology.

(1)Agent [27].

Against the backdrop of the continuous evolution of artificial intelligence technologies, LLMs [9], as important achievements in the field of natural language processing, have achieved in-depth understanding and generation of the complex semantics and grammatical structures of human language through the Transformer architecture and training on massive text data. On this basis, agents [27] built based on LLMs achieve functional upgrades and task expansions through modular design, with their core modules including memory modules, planning modules, tool modules, and action modules. This modular architecture enables agents to autonomously perceive, make decisions, and execute tasks in dynamic environments, significantly enhancing the task-processing capabilities and environmental adaptability of artificial intelligence systems. This study utilizes the multifunctional characteristics of agents to achieve specific task objectives. Aiming at the execution deviations and efficiency bottlenecks that may occur when agents process complex tasks, the research focuses on achieving precise definition of agent functions and behavioral guidance by optimizing prompt design and CoT [31] strategies, introduces RAG [271228] technology to address the hallucination problem, and meanwhile establishes a complete library of

Table 2
Table of Definitions and Meanings of Concepts[29] in AutoBEE.

Concept	Original Meaning	Meaning in This Study
Super Graph	A complex graph structure in theories, used to model system architectures.	The top-level framework in this research, coordinating task planning and supervision. Manages "Planning" and "Supervision" teams to ensure task execution alignment.
Sub Graph	A subset of a graph, enabling modular analysis.	Represents specific task stages (e.g., parameter modification, simulation, data analysis) within the Super Graph.
Team	A collaborative group working towards a common goal.	Specialized groups of agents/LLMs handling distinct roles (e.g., planning, simulation, reporting).
Nodes	Basic units in graph theory, representing entities or operations.	Individual agents or LLMs performing specific tasks (e.g., semantic parsing, simulation execution).
Edges	Connections between nodes, denoting relationships or data flow.	Define task dependencies and information transfer paths (e.g., parameter modification must precede simulation).
super nodes	Nodes with elevated management capabilities in distributed systems.	LLM-powered entities that dynamically plan execution paths based on real-time task status, enhancing adaptability.
Path Selection	Algorithmic process to determine optimal routes between nodes.	Dynamic decision-making by Super Nodes to select execution paths based on task complexity and resource availability.

building performance analysis tools.

(2) Chain of Thought (CoT) [31].

CoT refers to a method that decomposes complex tasks into a sequential series of reasoning steps. By simulating human step-by-step reasoning, CoT guides agents to generate reasoning and calculations based on previous results, ultimately leading to the achievement of the task objective.

The implementation of CoT primarily involves two approaches: prompt engineering and model fine-tuning. Prompt engineering entails designing guiding statements or examples within input instructions to direct the agent in constructing a reasoning chain. Model fine-tuning, on the other hand, involves training the agent on datasets annotated with CoT data to enhance its ability to autonomously generate reasoning steps.

In this study, prompt engineering is employed by embedding structured guiding statements and examples into input prompts, explicitly instructing the agent to analyze problems and plan tasks following specific logical steps. For example, task instructions may require the agent to "analyze the problem step-by-step: first, identify key elements; then, derive intermediate logic; and finally, draw conclusions." This approach enhances the agent's understanding and execution of tasks, thereby improving the accuracy and efficiency of task completion.

Here is an example of CoT prompt engineering applied to the modify agent:

You are an agent responsible for modifying IDF files. Your task is to follow the steps below using the {Chain of Thoughts} strategy:

(a) Receive the task instruction, comprehensively analyze the current system status, the existing information of the IDF file, and the specific modification requirements, and clarify the task objective.

(b) According to the task requirements, screen the suitable tools from the tool library and sort out the parameters and instructions required for tool invocation.

(c) Execute the tool invocation, input the parameters and instructions accurately, and start the modification operation on the IDF file.

(d) Monitor the tool execution process in real − time, obtain the feedback result of the modification operation, and determine whether the modification is successful.

(e) If the modification is successful, jump to step 8; if it fails, check the input parameters and instructions of the tool, and optimize and adjust them in combination with the feedback error information.

(f) Invoke the modification tool again, execute the modification of the IDF file, and verify the modification result again.

(g) Repeat steps e − f until the modification is successful.

(h) Output the prompt message indicating that the modification of the IDF file is successful, and confirm that the modified content is correct.

(i) Save the modified IDF file according to the specified path and format to complete the task.

(3) Tools library for building performance analysis.

The tools of an agent are components that assist the agent in completing specific tasks and expanding its capabilities. These tools come in various forms. For example, a search engine can provide real-time information, a code interpreter can execute code to solve programming problems, a file reader allows access to and processing of local files, and a calculator can perform numerical calculations, among others. These tools function as the "assistants" of the agent. By breaking down complex tasks into calls to different tools, the agent can extend its capabilities and efficiently complete a wide range of tasks, such as data analysis, knowledge retrieval, and content creation, significantly improving the efficiency and quality of problem-solving.

To enhance the comprehensiveness and efficiency of agents in executing tasks in the field of building performance analysis, this study has established a complete tool library covering all dimensions of IDF file modification, simulation execution, and report analysis, specifically including IDF file modification from perspectives such as parameters of enclosure structure, building morphology and spatial layout, operation parameters of equipment systems, behaviors of personnel and equipment, and multi-type pollutant emission factor as well as aspects like EnergyPlus API invocation, building energy consumption analysis, indoor environmental parameter analysis, and pollutant emission analysis, which can be widely applied to the whole process of BECPA with excellent versatility. In this study, the construction of tools in the tool library mainly adopts three invocation modes, as shown in Fig. 2. In mode (a), the LLM directly invokes encapsulated custom Python tools after planning. These tools are presented as functions, with the model passing parameters for execution and receiving feedback on the results. In this process, strategies such as reflection, self-criticism, CoT, and subgoal decomposition assist in the planning phase. In mode (b), the LLM first plans and generates executable Python code, which is then run in a Python execution environment, with the running results being fed back to the model. Again, the aforementioned auxiliary strategies are employed. In mode (c), the LLM directly invokes non-custom tools, such as text chunking, text vectorization, vector storage, and summarization, with the results being returned after the tools are executed. Strategies like reflection are also used to support the planning process. These three modes illustrate the flexibility and adaptability of LLMs in utilizing tools to complete tasks in this study.

The tool invocation method shown in mode (c) is primarily applied to RAG technology. This approach effectively addresses the "hallucination" problem often encountered by large models when handling specialized knowledge. By combining the retrieval process from an external knowledge base with the generative capabilities of LLMs, RAG ensures more accurate and reliable outputs. In the field of building performance analysis, users' core demands are frequently closely tied to industry standards and specifications. However, when LLMs answer questions involving standard clauses and compliance requirements, errors or fictional content may emerge. To address this issue, this study develops a targeted application scheme for RAG. First, authoritative standard documents, such as building energy consumption calculation standards and indoor environmental quality specifications, are vectorized to create a specialized domain knowledge base. Simultaneously, user input questions are transformed into vector representations. Using algorithms such as cosine similarity, semantically relevant standard knowledge fragments are retrieved from the knowledge base and input into the LLM as context. This mechanism ensures that the answers generated by the LLM are consistently based on authoritative standards, effectively mitigating the risk of "hallucination" and enhancing the professionalism and credibility of the building performance analysis conclusions produced by the report team. Examples of the specific implementation methods for these three tools are detailed in Appendix A.

Through the above strategies and the constructed tool library of BECPA, this study has improved the performance of a single agent. Meanwhile, the tool library can be extended to other application scenarios of BECPA and has universality.

2.2.2. *Agent groups collaborative capability*

(1) Multi-level network composed of teams and agents.

When handling complex building performance analysis tasks, traditional single agent or centralized processing models often face efficiency bottlenecks and frequent errors. Breaking down tasks and distributing them across multiple teams, where multiple agents collaborate within each team, has proven to be an effective strategy for enhancing task processing efficiency [29]. Therefore, AutoBEE is designed as a multi-tiered framework. Agents with similar performance capabilities are first grouped into teams, including the Planning Team, Supervision Team, Modification Team, Simulation Team, and Report Team, which further compose the entire framework. This approach ensures that tasks go through the process of being allocated to teams and then to agents, which significantly enhances the collaborative capabilities of agent groups. (2) to (4) will introduce the main technologies of this multi-level

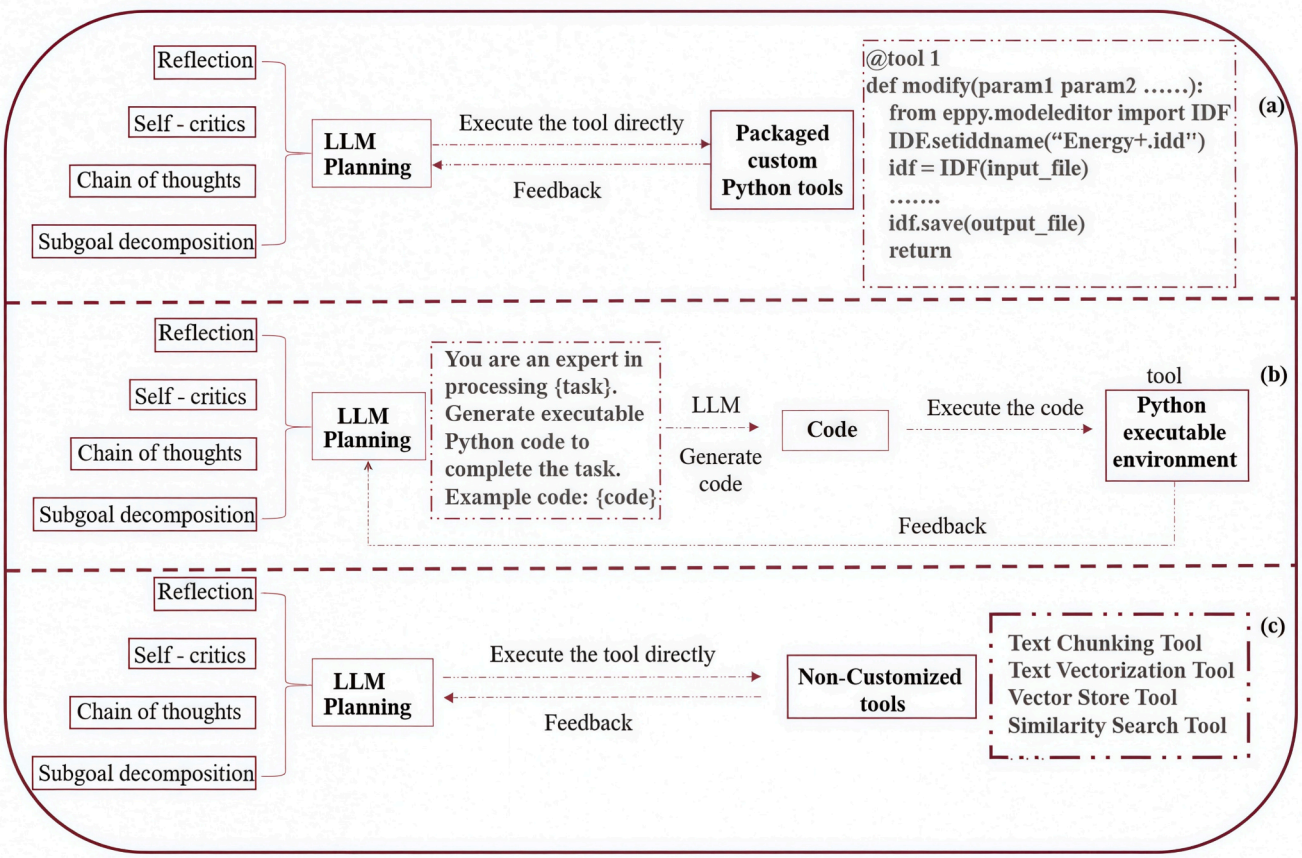


Fig. 2. Three Tool Invocation Methods of the Agent.

framework design, and (5) will demonstrate the teams created based on these technologies and the capability enhancement of individual agents.

(2) LangGraph [3229].

LangChain is a framework specifically designed for developing applications driven by LLM [29]. It provides a series of tools, components, and interfaces to help developers efficiently connect with and invoke large language models. This not only simplifies the integration process of models with external data sources and tools but also supports the construction of complex application logic. As an extension of LangChain [29], LangGraph focuses on building powerful stateful multi-role applications. It models the application process as a graph structure, where nodes represent operational steps and edges represent the relationships between steps. This enables efficient orchestration of multi-agent collaborative tasks, provides an interface for framework design, and significantly enhances the logic and traceability of task planning and execution. Meanwhile, the stateful feature of LangGraph allows agents to retain and utilize the results of previous steps during execution, supporting context-based dynamic decision-making. The definitions of concepts related to LangGraph used in this study are shown in Table 2.

(3) Multi-level super node application [29].

In multi-agent collaborative scenarios for complex task execution, team collaboration capabilities are critical, with agent scheduling rationality directly determining task progression efficiency. Failure to accurately allocate tasks to corresponding agents based on real-time status, or information mistransmission during allocation, will inevitably lead to complex task execution failures. Specifically, the absence of global oversight causes task fragmentation: agent groups lacking centralized control often incur global imbalances due to locally optimal decisions, such as energy consumption simulation agents and indoor environment analysis agents competing for computing resources via parallel calls, thereby stalling tasks. Meanwhile, blind path selection triggers efficiency collapse: in multi-stage tasks involving IDF file modification, EnergyPlus simulation, data validation, agents relying solely on local rules to autonomously select execution paths may enter cyclic loops or omit critical steps, such as initiating simulations before parameter optimization is complete and yielding invalid results.

To address the scheduling issues, this study introduces multi-level super nodes based on the design flexibility of LangGraph. After each team or agent completes its assigned task, it returns the task completion status to the super node composed of LLMs. The super node then generates instructions for the next task based on global optimality and the current state, generate instructions for subsequent tasks based on global optimality principles, and macro-regulate the path planning for task completion. Specifically, at the overall BECPA task level, super nodes select the next execution team according to global optimal strategies. For example, when a user only needs to simulate the current IDF file and generate a report, AutoBEE will invoke the (1) Simulation Team and (2) Report Team. While when a user requires scenario comparison, the system will sequentially call the (1) Simulation Team, (2) Modification Team, (3) Simulation Team (re-invocation), and (4) Report Team, The simulation team will not mistakenly assume that the simulation task has been completed due to a previous invocation. At the intra-team level, sub super nodes are responsible for selecting the next execution agent. If user only needs to modify parameter A, the modification team's supervisor will assign the task exclusively to agent A, or if parameter B needs to be modified after A, the supervisor will first allocate the task to agent responsible for modifying A and then to the agent responsible for modifying B. Through this centralized regulation mechanism, issues such as locally optimal path selection, cyclic calls, and critical step omissions can be effectively avoided. Additionally, the configuration of Super Nodes facilitates communication among agent groups, a feature that will be elaborated in section (3).

(4) Lightweight communication protocol.

In multi-agent collaboration scenarios, flexible configuration of

communication strategies is crucial for ensuring system performance and controlling operational costs [30]. When executing complex tasks, each subtask completion at both the team and individual agent levels generates a large amount of status updates and information transmission. If all information is passed to the next task executor without filtering, it will not only significantly increase the token consumption of LLMs but also interfere with agent decision-making due to information overload, remarkably increasing task execution difficulty and even leading to overall task failure. To address this, this study leverages LangGraph's capability to allow developers to customize message transmission paths and rules based on task requirements, designing a lightweight communication mechanism. This mechanism implements fine-grained control over status information through a hierarchical architecture of hypergraphs and subgraphs. The super node in the hypergraph serves as a global coordination hub, aggregating and distributing only the core input statuses required by subgraph teams. Each subgraph team, after completing tasks, integrates task status via sub super node and feeds back key results to the super node. Specific status data during task execution are strictly confined within team boundaries, with execution details never shared across teams. For example, as shown in the example of Fig. 3, state 1 is the information aggregated by the super node based on the current task status and intended for team 1. after the information is transmitted to team 1, it is managed by the sub super 1 node to carry out task completion and information transfer (state 1.1 to state 1.6) between internal nodes of the team. finally, after team 1 completes the task, the sub super 1 node filters out the intermediate states of the task, consolidates the task completion status, updates state 2, and transmits it to the super node. subsequently, the super node communicates with team 2 in the same manner. thus, states 1.1 to 1.6 are only shared within team 1, while team 2 can only receive information about team 1′s task completion and not the details thereof.

This communication strategy offers dual advantages. On one hand, by reducing invalid information transmission, LLM token consumption is significantly minimized, directly cutting operational costs; On the other hand, by precisely filtering irrelevant information, agents can focus on core tasks, avoiding decision-making interference. This communication setup not only ensures efficient collaboration among agents but also optimizes overall system performance, providing a reliable foundation for the efficient execution of complex tasks.

(5) Multi-agent team [5112529].

Based on the aforementioned capability enhancement of individual agents, team design, and basic communication technologies, this study has established the following teams, as shown in Fig. 4. And the operation of the overall framework can be seen in Fig. 1.

(a) Planning Team: This team primarily leverages the text processing capabilities of LLM. Based on building performance analysis-related issues proposed by users, such as requirements for energy consumption and indoor environment, it generates comprehensive and detailed task plans. By performing in-depth semantic understanding and logical analysis of the issues, the team clarifies the task arrangements for each stage and the division of labor among agents, thus providing a solid foundation for the smooth execution of subsequent tasks.

(b) **Supervision Team**: This team conducts comprehensive supervision of the task execution process and manages the communication and collaboration among team members. This team dynamically plans the next execution direction of the project in real-time based on the feedback of the execution status.

(c) **Modification Team**: Multiple agents within the team are equipped with corresponding tools (mainly adopting the tool execution methods shown in Fig. 2 (a). For the task of modifying parameters in the IDF file in building performance analysis, each agent gives full play to their expertise. Meanwhile, sub-supervisor nodes are set within the team for internal path planning of the team. According to the feedback of the task execution status, the sub-supervisor nodes flexibly dispatch agents and tools to ensure the accurate and efficient completion of the parameter modification work.

(d) **Simulation Team**: Although the task is relatively simple, it has extremely high requirements for professionalism and accuracy. Agents within the team are equipped with various tools (mainly adopting the tool execution methods shown in Fig. 2 (a), focusing on conducting simulation calculations on the modified IDF file and preliminarily processing the simulation results, converting them into a format convenient for analysis. Due to the clear characteristics of the task, no super nodes

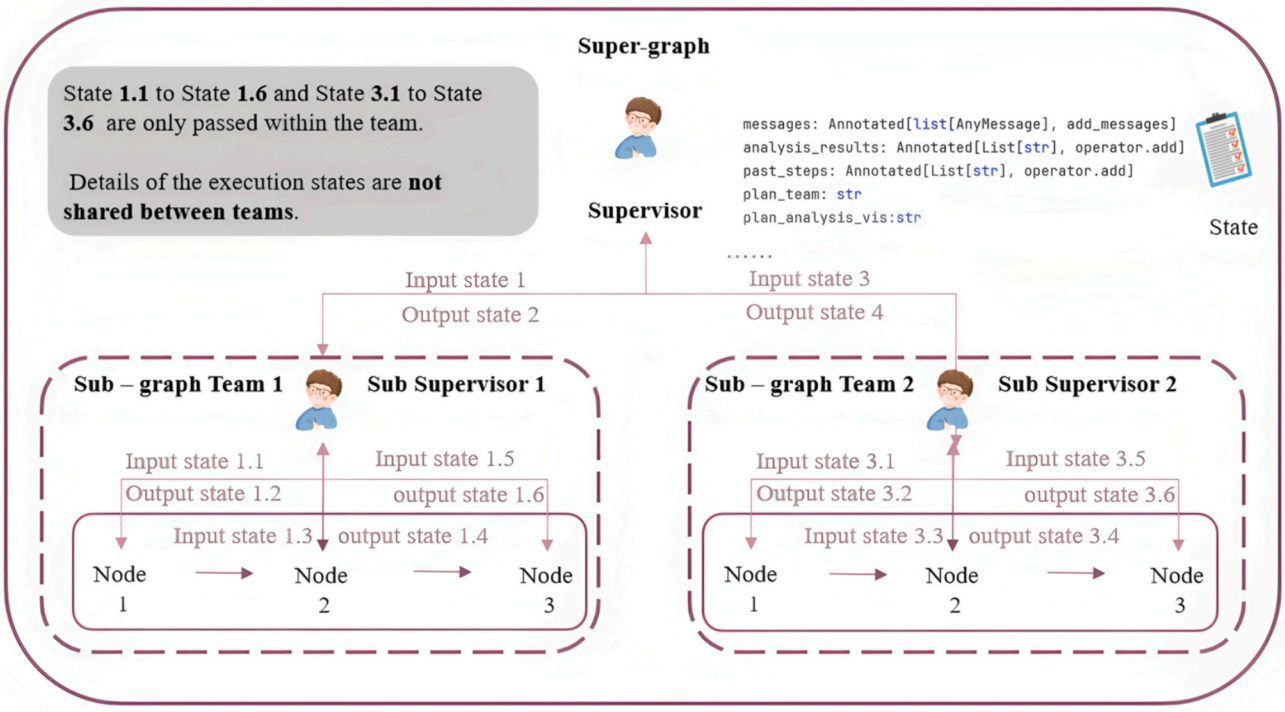


Fig. 3. State Transmission.

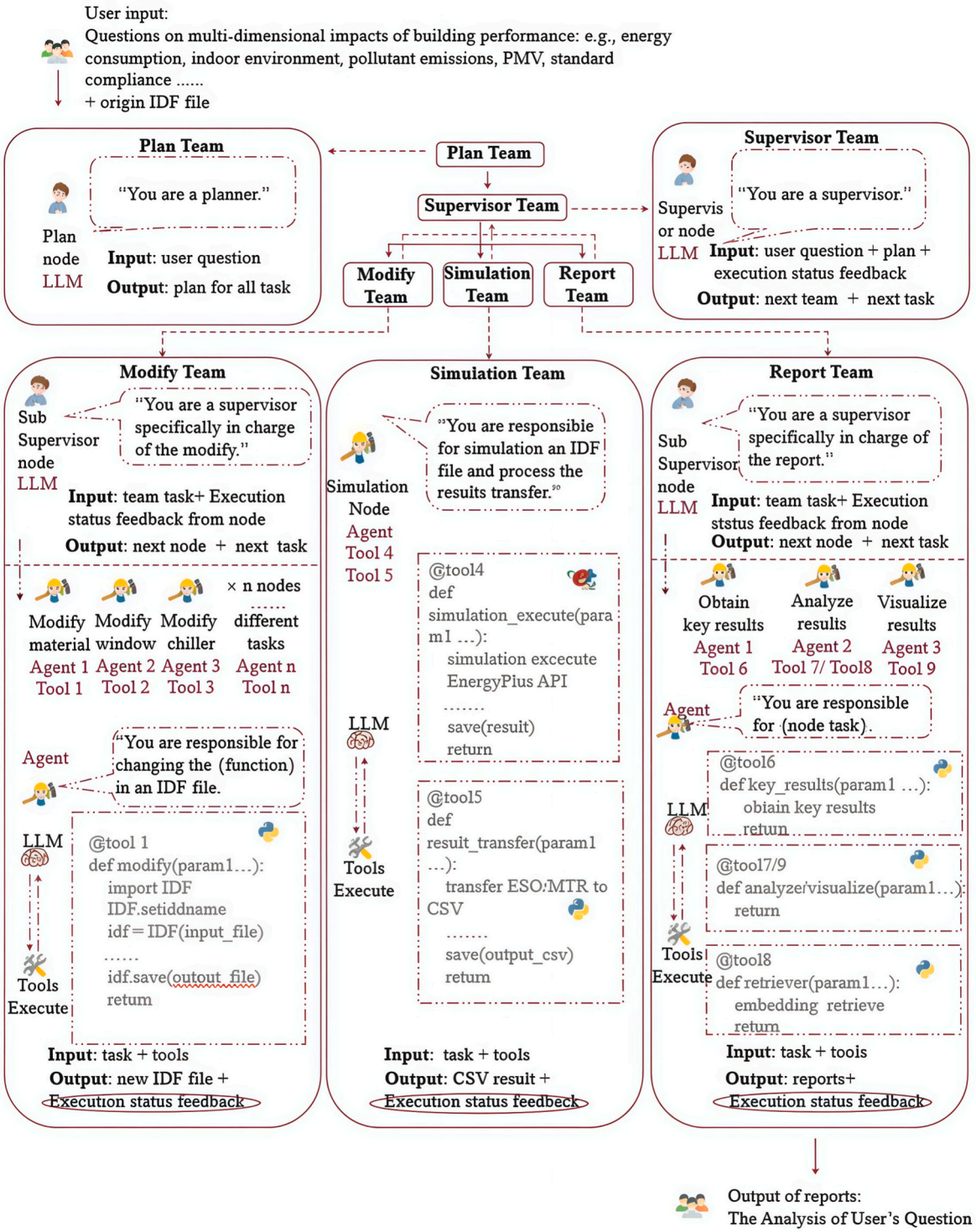


Fig. 4. Multiple teams of AutoBEE.

are set in this team to simplify the process and improve execution efficiency.

(e) **Report Team**: Multiple agents within the team are equipped with various tools (mainly adopting the tool execution methods shown in Fig. 2 (b) and (c). They are responsible for comprehensively collecting, deeply analyzing, and visually presenting the data generated by the simulation. At the same time, sub-supervisor nodes are set for internal path planning of the team. According to the task progress and data characteristics, agents and tools are reasonably allocated. In addition, this team uses RAG technology to introduce external standard data and knowledge, effectively reducing the hallucination problems that may occur when the large model generates reports, ensuring that the finally output analysis reports are professional, accurate, and reliable.

The operation of each team follows the methods mentioned in (2) and (3). And the key prompts for all nodes within each team are provided in Appendix B.

2.2.3. Mathematical model of AutoBEE

As synthesized from the above mentioned content, the mathematical model of AutoBEE for task execution is as follows: Formula a represents the operating program of the framework, Formula b represents the lightweight information passing strategy among teams, and Formula c represents the dynamic path planning strategy led by the super node.

$$Y_{\text{AutoBEE}} = F_{\text{Plan}}(F_{Supervise}(\sum_{k=1}^{K} F_{Sub-Supervise,k}(\sum_{i=1}^{N_k} A_{k,i}))) (a)$$

$$I_{Sub-Supervise,n} = Ç¶^{m}_{hide}\left(\sum_{i=1}^{N_m} O_{m,i}\right),\ m,\ n \in ¼,\ m \neq n\ (b).$$

$$F_{Supervise/Sub-Supervise} = \underset{P_t \in P}{\operatorname{argmin}}\left(\sum_{(i)\in P_t} C_i(\hat{E}_{current})\right)(c).$$

- Y_{AutoBEE}: Structured professional analysis report finally output by the AutoBEE system.
- F_{Plan}: Planning-layer function that decomposes user tasks into an initial execution plan.
- $F_{\text{Supervise}}$: Super-node function monitoring and adjusting the entire task execution.
- $F_{\text{Sub-Supervise,k}}$: Sub-supervision function for the k_{th} subtask team ($k = 1,\ldots,K$).
- $A_{k,i}$: Atomic action i of subtask k from agent ($i = 1,\ldots,N_k$).
- $I_{\text{Sub-Supervise,n}}$: Information to the sub-supervisor of team n.
- $Ç¶^{m}_{hide}$: Function for processing hidden information.
- $O_{m,i}$: Original information generated by the i_{th} agent in team m ($i = 1,\ldots,Nm$).
- P: Set of all possible paths or execution sequences.
- $ɖ_{\text{current}}$: Current state of the system or task execution environment.
- C_i: Execution cost of the path i.

Formula (a) indicates that the AutoBEE framework is a collaborative framework involving multiple teams and multiple agents. Formula (b) shows that detailed information within teams is not shared between teams. Formula (c) demonstrates that super nodes at all levels dynamically plan the way tasks are completed based on the current state and path costs.

3. Experiments and results

3.1. Experimental environment

The experiments were conducted using Python 3.11.11 with PyCharm as the development environment. A multi-agent system was built upon the LangGraph framework, integrating GPT4o as the LLM to provide support. This study focused on the accuracy and stability of results rather than their diversity and divergence. Therefore, the hyperparameters of Temperature and top-p were both set to 0.2. Meanwhile, all input original IDF files were sourced from the EnergyPlus official website.

3.2. Experiments

In the experimental environment constructed above, this study has established an architecture for multi-agent collaborative work. The following presents two cases, aiming to verify the effectiveness and practicality of the multi-agent framework in tasks of analyzing the impacts of energy technology measures on energy consumption and indoor environment through real cases. Each case focused on specific building performance analysis issues. By utilizing the collaborative operation of teams in the multi-agent framework, starting from the problem raised by the users' natural language, the professional analysis report for the problem was finally outputted. The cases clearly demonstrated the process and advantages of the multi-agent framework in handling complex building performance analysis tasks.

3.2.1. Case 1: Impact of the COP improvement of the central air conditioning unit on energy consumption

Case 1 as shown in Fig. 5, demonstrated the effectiveness of the multi-agent framework in addressing performance analysis issues related to building energy consumption. At the beginning of the experiment, the planning team received the user's question about the reduction in energy consumption after adjusting the coefficient of performance (COP) of the central air conditioning unit to 4.0, and also obtained the original IDF file. Through the in-depth semantic analysis and task decomposition capabilities of the LLM, the planning team formulated a detailed execution plan and clarified the subsequent workflow of each team as shown in Fig. 5(b) state 1. The simulation team took the lead in conducting an initial simulation using the original IDF file. Relying on the simulation tools integrated with agent, the team generated complete energy consumption data under the baseline condition and output it as standard format files such as ESO file. These data provided an important reference for subsequent comparative analysis. After completing the initial simulation, the modification team, under the instructions of the supervision team, precisely adjusted the parameters of the IDF file. The team focused on the "central air conditioning unit" component in the file and accurately modified its COP value to 4.0. Subsequently, the simulation team conducted a second round simulation based on the modified IDF file, aiming to obtain the energy consumption data after adjusting the COP value. The result file generated by the simulation contained key information about the changes in energy consumption and was transmitted to the reporting team. The reporting team conducted a systematic comparative analysis of the results of the two simulation rounds. By extracting the core energy consumption indicators and applying statistical methods, the team generated a detailed energy analysis report. In this experiment, the entire process was effectively monitored with the help of the LangSmith [33] monitoring platform. The operation steps are shown in Fig. 5a, and the status update is presented in Fig. 5b. The finally output files and reports are shown in Fig. 5c. Specific details of the report can be found in Appendix C.

The energy analysis summary report was rich in content, covering detailed energy consumption data in the two scenarios, including the monthly electricity consumption of facilities, the monthly electricity consumption of buildings, the monthly electricity consumption of HVAC systems, and the monthly electricity consumption of heating/cooling plant. It focused on presenting the comparative analysis results. After modifying the COP value of the central air conditioning unit to 4.0, the total electricity consumption at the facility level was reduced by approximately 2.10 %, and the electricity consumption at the power plant level was reduced by approximately 17.43 %. The report not only quantified the reduction in energy consumption but also provided a scientific basis for evaluating the energy saving benefits of the COP adjustment, directly responding to the user's initial question. The entire process took 92 s, and after manual verification, the results were completely correct. Case 1 comprehensively demonstrated the structured, efficient, and accurate advantages of the AutoBEE framework in handling complex building performance analysis tasks.

3.2.2. Case 2: Influence of the number of people per unit area on PMV

Case 2, as shown in Fig. 6, demonstrated the effectiveness of the multi-agent framework in dealing with building performance analysis issues related to indoor thermal comfort. At the beginning of the experiment, the user raised the question of "Whether the PMV requirements of ASHRAE 55 standard can be met when the number of people per square meter in the indoor environment increases by 2″ and provided the original IDF file. The planning team, through the semantic analysis ability of the LLM, quickly disassembled the task and formulated an execution plan, clarifying the workflow of the modification, simulation, reporting and other links. The modification team made targeted adjustments to the relevant parameters in the IDF file according to the task instructions, providing basic data that met the experimental settings for subsequent simulations. The simulation team, based on the

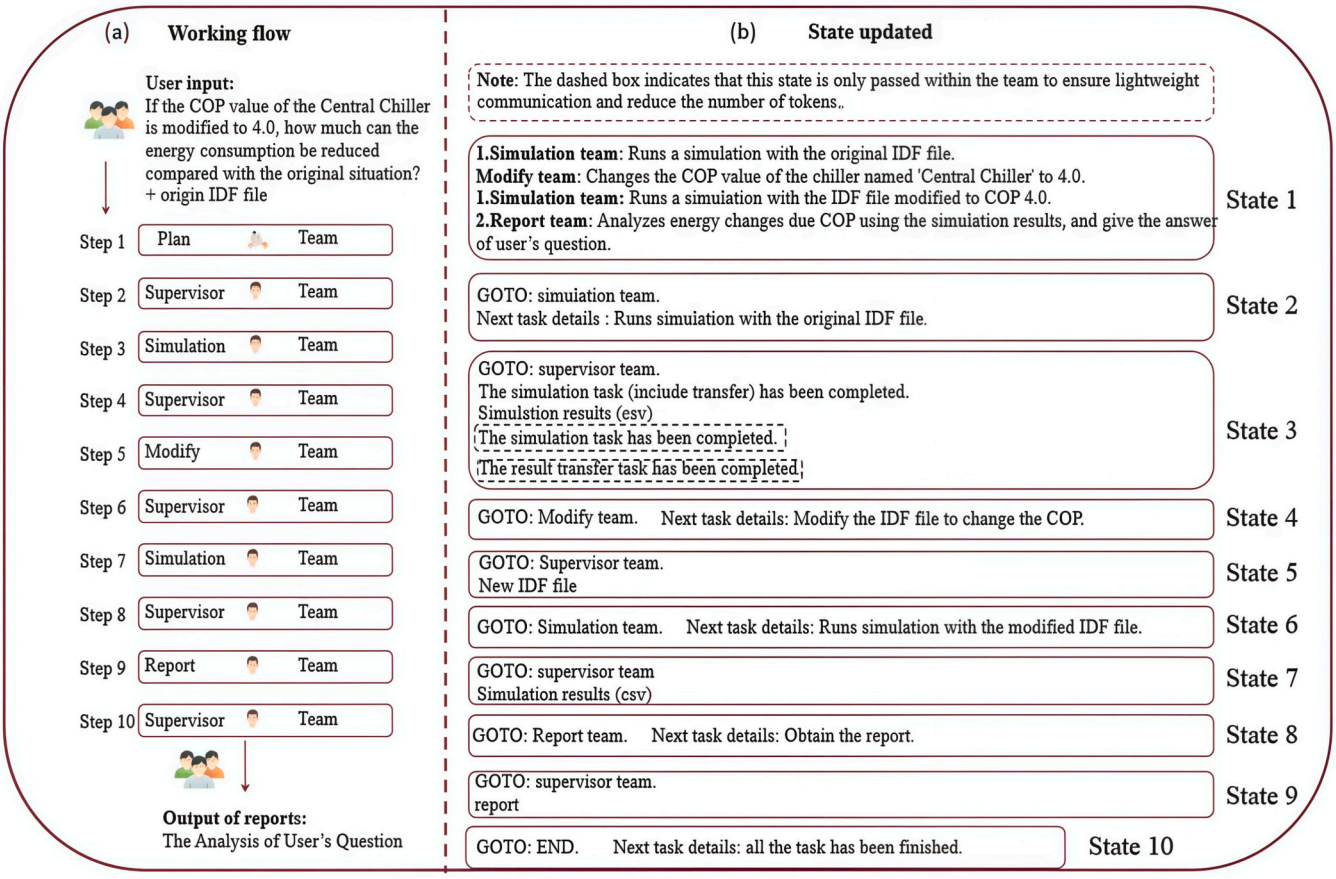


Fig. 5. Impact of the COP improvement on energy consumption.

modified file, used the simulation tools integrated in the agent to simulate the indoor thermal environment and obtained simulation result files containing key data such as temperature, humidity, and PMV values. The reporting team was responsible for conducting in-depth analysis of the simulation data and generating the final report. The information transfer and status update mechanism among teams was consistent with that in Case 1: Supervision teams at all levels coordinated task scheduling, distributed instructions, and allocated resources based on the task completion status of each team or agent. Each team transmitted result files through standardized data interfaces, and internal status updates of the teams were not shared externally.

Taking the reporting team as an example, its internal workflow showed rigorous hierarchical characteristics. First, the reporting supervision node (Step 2.1) received the task instruction and confirmed that key data needed to be extracted from the simulation results, and the status was updated to "Task received and to be processed" (State 2.1). Subsequently, it entered the data extraction stage (Step 2.2), and the team screened out core indicators such as the distribution of PMV values and temperature and humidity parameters from the CSV file, and the status was updated to "Key data obtained" (State 2.2). In the pre-analysis link (Step 2.3), the supervision node intervened again to clarify that compliance analysis needed to be carried out in combination with the ASHRAE 55 standard, and the status was changed to "Analysis ready" (State 2.3). Entering the core analysis stage (Step 2.4), the reporting team used the RAG technology for double verification. First, they vectorized the user's question and simulation data, conducted semantic retrieval in the ASHRAE 55 standard knowledge base (Step 2.4.1), and matched relevant clauses such as "The influence of the increase in personnel density on the PMV threshold" through the cosine similarity algorithm. Then they conducted a comparative analysis of the simulated PMV data based on the retrieval results (Step 2.4.2) to determine whether it met the standard requirements, and the status at this stage was updated to "Analysis and retrieval completed" (State 2.4.2). Finally, the supervision node (Step 2.5) confirmed that all analysis tasks were closed, and the status was updated to "Report generated and ready for output" (State 2), and the energy analysis report containing the statistical distribution of PMV values, standard compliance conclusions and optimization suggestions was delivered to the next link. Meanwhile, according to the information hiding strategy mentioned in 2.2.2, all statuses from 2.1 to 2.5 will not be transmitted to other teams. This process not only ensured the authority of the analysis results but also effectively avoided the "hallucination" problem of the large model through the RAG mechanism.

Through the LangSmith monitoring platform, the operation steps and status update were shown in Fig. 6a, and the finally output files and reports were shown in Fig. 6c. Specific details of the report can be found in Appendix C.

The report provided the occurrence frequencies of different PMV value intervals. The PMV value appeared 191 times between −3 and −2, 41 times between −2 and −1, etc., Most of the PMV values were distributed in the intervals of −3 to −2 and 2 to 3. The result summary and question answering section indicated that according to the ASHRAE 55 standard, the ideal range of PMV values is between −0.5 and 0.5. The analysis showed that most PMV values exceeded this range, indicating the current indoor environment did not satisfy thermal comfort requirements, with conditions being either excessively cold or hot, necessitating environmental adjustments to meet the standard. The entire process took 61 s, and after manual verification, the results were

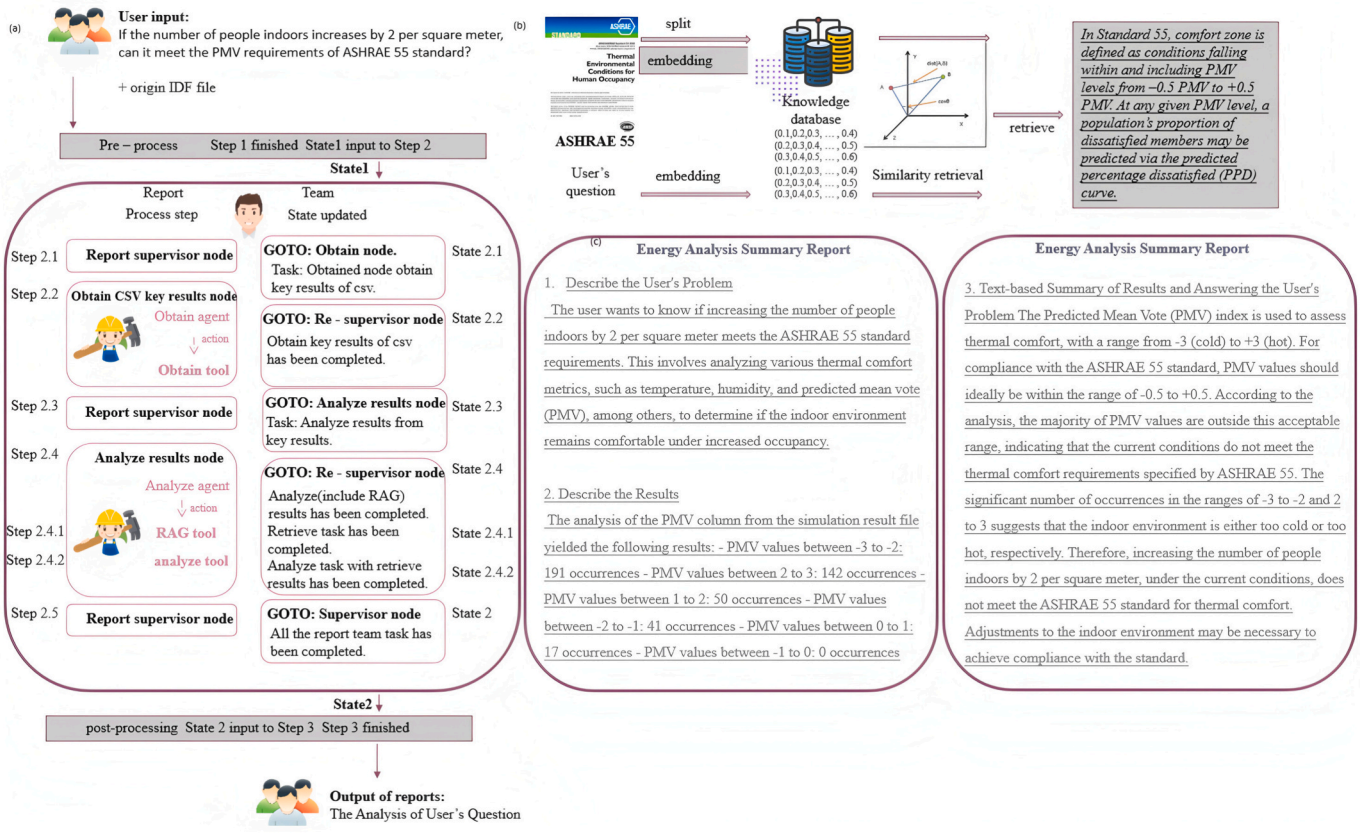


Fig. 6. Influence of the number of people per unit area on PMV.

completely correct, fully demonstrating the efficiency and accuracy of the multi-agent framework in addressing such challenges.

3.3. Validation across all dimensions of building performance analysis

3.3.1. Typical application case collection of AutoBEE

To assess the accuracy and robustness of the framework, this study systematically collected 54 typical application scenarios that engineers frequently encountered in building performance analysis. The specific cases were detailed in Appendix D, under "Question Description." The core design logic for these questions followed a key principle: all relevant dimensions were fully covered, with one representative question assigned to each single dimension. Specifically, for each category of questions, the study selected one representative scenario to conduct analysis. Take the building orientation issue as an example. In practical engineering, architects usually only needed to modify the orientation parameter in the IDF file, and the corresponding scenario type was relatively simple. Therefore, the study could summarize this type of engineering scenario with just one question. For objects like chillers and cooling water systems, however, the related issues involved multiple technical dimensions such as COP and pumps. For this reason, the study also selected one specific scenario for analysis corresponding to each of these dimensions. These 54 questions comprehensively covered the core dimensions of building performance analysis. By inputting these complex, multi-dimensional questions into the framework, its accuracy and stability in core tasks such as building energy consumption prediction, environmental parameter analysis, and performance standard evaluation were thoroughly tested. This process verified the framework's ability to meet the practical needs of engineering applications. Table 3 presented the classification of the verification cases for AutoBEE, where the "Serial Number" column corresponded directly to the question numbers in Appendix D.

Table 3
Classification of Validation Cases for AutoBEE Framework.

Research Dimension	Variable Parameters	Evaluation Direction	Question number
Parameters of enclosure structure	Heat transfer coefficient		1, 2, 3, 4, 35, 47
Building morphology and spatial layout	Building orientation		32
	Window-wall area ratio		5, 36
	Exterior window orientation		6, 37
	building shading system		7, 8, 38
	Room infiltration rate		9, 39
Operation parameters of equipment systems	Set values of air supply		14, 15, 41
	Temperature control points		16, 17, 33
	fans and pumps		20, 21, 22, 23
	COP and EER		26, 30, 31
	Cooling tower		18, 19
	Chilled/cooling water		24, 25, 27, 28, 29
Behaviors of personnel and equipment	Distribution of personnel density		10, 13, 34, 46, 48
	indoor electrical equipment		11, 12, 40
Multi-type pollutant emission factor	Electricity emission factor	Carbon emissions and pollutant emissions	42, 43, 44, 45
/	/	Standard (ASHRAE 55, GB50736 et al.)	49 ~ 54

3.3.2. Comparative experimental design and evaluation criteria

To verify the effectiveness of the AutoBEE framework compared to traditional methods, this study conducted a comparative experiment. Three postgraduate students, each with a comprehensive understanding of building environments and HVAC systems, as well as experience using the EnergyPlus software, were recruited as testers. Two questions from each of the six categories in the question bank, along with the relevant standards, were provided to the testers. Each participant was required to independently complete the analysis task based on their professional knowledge, using the EnergyPlus software, and generate a report that included key data records, important analysis conclusions, and direct answers to the questions. The time taken for each task, from the issuance of the question to the submission of the report, was accurately recorded. For the AutoBEE framework, these 12 application scenarios were also inputted, and the framework automatically generated analysis reports according to the established process. To ensure the objectivity and reliability of the evaluation, a senior designer with extensive experience in building performance analysis was invited to serve as the evaluator. It is important to note that AutoBEE was run three times repeatedly for each question. The evaluator first needed to check whether there were differences in numerical values (e.g., energy consumption values) and core judgment items of the report (e.g., compliance with a specific standard) among the three runs. If differences existed, the evaluation was terminated immediately and relevant circumstances were recorded. If no differences existed, the next step of evaluation was conducted. This process could effectively ensure the stability and reliability of the framework operation. On this basis, the evaluator conducted a comprehensive assessment of the reports generated by the AutoBEE framework and those from the three testers, with the evaluation criteria covering three dimensions: accuracy, rationality, and content richness.

In terms of accuracy, a binary judgment method was applied. If the data values and calculation results in the report fully aligned with the actual situation, it was classified as "accurate"; otherwise, it was considered "inaccurate." For rationality and content richness, both the manually generated reports and those output by the AutoBEE framework were scored on a scale of 10. The rationality score focused on the logical coherence of the analysis process and the applicability of the method, while the content richness score considered the integrity of data presentation, the depth of the analysis, and the comprehensiveness of the solution. After collecting all the evaluation data, the average scores of the manually generated reports in the dimensions of rationality and content richness were calculated and compared with those of the AutoBEE framework. If the score of the AutoBEE framework was higher, the final comparison value was recorded as 1; if both scores were the same, it was recorded as 0; if the AutoBEE framework score was lower, it was recorded as −1. A double-blind method was employed during the evaluation process, meaning the evaluator was unaware of the source of the reports (whether from the AutoBEE framework or manual analysis) in order to eliminate potential subjective biases. This comparative experimental design systematically and objectively compared the AutoBEE framework with traditional manual analysis methods, providing empirical evidence to demonstrate the framework's advantages in terms of efficiency, accuracy, and analysis quality.

3.3.3. Experimental design for information management strategy

To verify the effectiveness of the design of achieving lightweight communication among multi-agent through specific information management strategies and controlling the token consumption of large models by hiding partial detailed information in improving task completion and economic efficiency, an independent verification experiment was conducted. The experiment maintained the same programming environment and hardware configuration as in the previous research and selected 54 typical application scenarios of building performance analysis as the test cases. In this experiment, all agents were set to adopt a fully interconnected message mechanism, meaning no information hiding was implemented to ensure complete information sharing among agents. During the experiment, the system's execution of tasks for the 54 scenarios was strictly recorded. If a task was successfully completed, the total token consumption generated during the process was simultaneously tracked.

3.3.4. Experimental results

Regarding the task processing time, the time data of the AutoBEE framework was recorded in the column of "AutoBEE time / S" in Appendix E, and the time data of manual task processing was recorded in the column of "Manual Processing Time / S" in Appendix E. The accuracy evaluation results were presented in the column of "Report correctness" in Appendix E, where "correct" indicated accuracy and "incorrect" indicated inaccuracy. For the rationality and content richness, the comprehensive comparison results were shown in the columns of "Rationality Comparison Result" and "Content Richness Comparison Result" in Appendix E, using a quantification method of −1 (AutoBEE scores lower than manual processing), 0 (both scores are the same), and 1 (AutoBEE scores higher than manual processing). The economic costs of adopting specific information management strategies were shown in the column of "Cost with Information Hiding ($)" in Appendix E, and the economic costs of the full-intercommunication message mechanism were shown in the column of "Cost without Information Hiding ($)" in Appendix E.

In the comprehensive verification of typical application scenarios, in terms of BECPA, the AutoBEE framework performed excellently in terms of high efficiency, accuracy, rationality, content richness, and economy.

(1) High efficiency.

In terms of the processing efficiency of building performance analysis tasks, the AutoBEE framework demonstrated superiority. Fig. 7(a) presents the time distribution of AutoBEE's processing of 54 building performance analysis problems. The results showed that the processing time fluctuated within the range of 50 to 99 s, reflecting the high efficiency and stability of the framework when dealing with tasks of different complexity levels. Among them, for some difficult problems involving multi-parameter coupling analysis and complex logical reasoning, AutoBEE was still able to complete the full-process analysis in about 90 s, fully demonstrating its task processing ability. Fig. 7(b) shows the comparative experimental results of the processing time between AutoBEE and manual processing. When handling 12 identical building performance analysis problems, the time consumption of AutoBEE was significantly lower than that of manual processing. For example, when dealing with Problem 1, AutoBEE only took 90 s, while manual processing took as long as 1562 s, with an efficiency improvement of nearly 17 times. In Problem 38, the manual processing time was 2062 s, while AutoBEE reduced it to 80 s. This comparative experiment clearly showed that through multi-agent collaborative operation, dynamic task planning, and efficient invocation of professional tools, the AutoBEE framework increased the execution efficiency of building performance analysis tasks by 10 to 20 times, effectively solving the problems of cumbersome and time-consuming traditional manual analysis processes and providing an efficient and reliable automated solution for building performance research.

(2) Accuracy.

In terms of accuracy verification, this study focused on 54 problems covering multi-dimensional scenarios of building performance analysis. Through manual verification of each case, it was found that the analysis reports generated by the AutoBEE framework exhibited a high degree of accuracy in both numerical calculations and conclusion derivations. Moreover, the results obtained from repeated runs of the same task demonstrated stability. Based on precise parameter adjustments, rigorous simulation calculations, and reliable knowledge reasoning, AutoBEE was able to produce analysis results without numerical deviations, effectively avoiding the human calculation errors and logical inconsistencies that were prone to occur in traditional analysis models. This provided a trustworthy analytical foundation for building performance research.

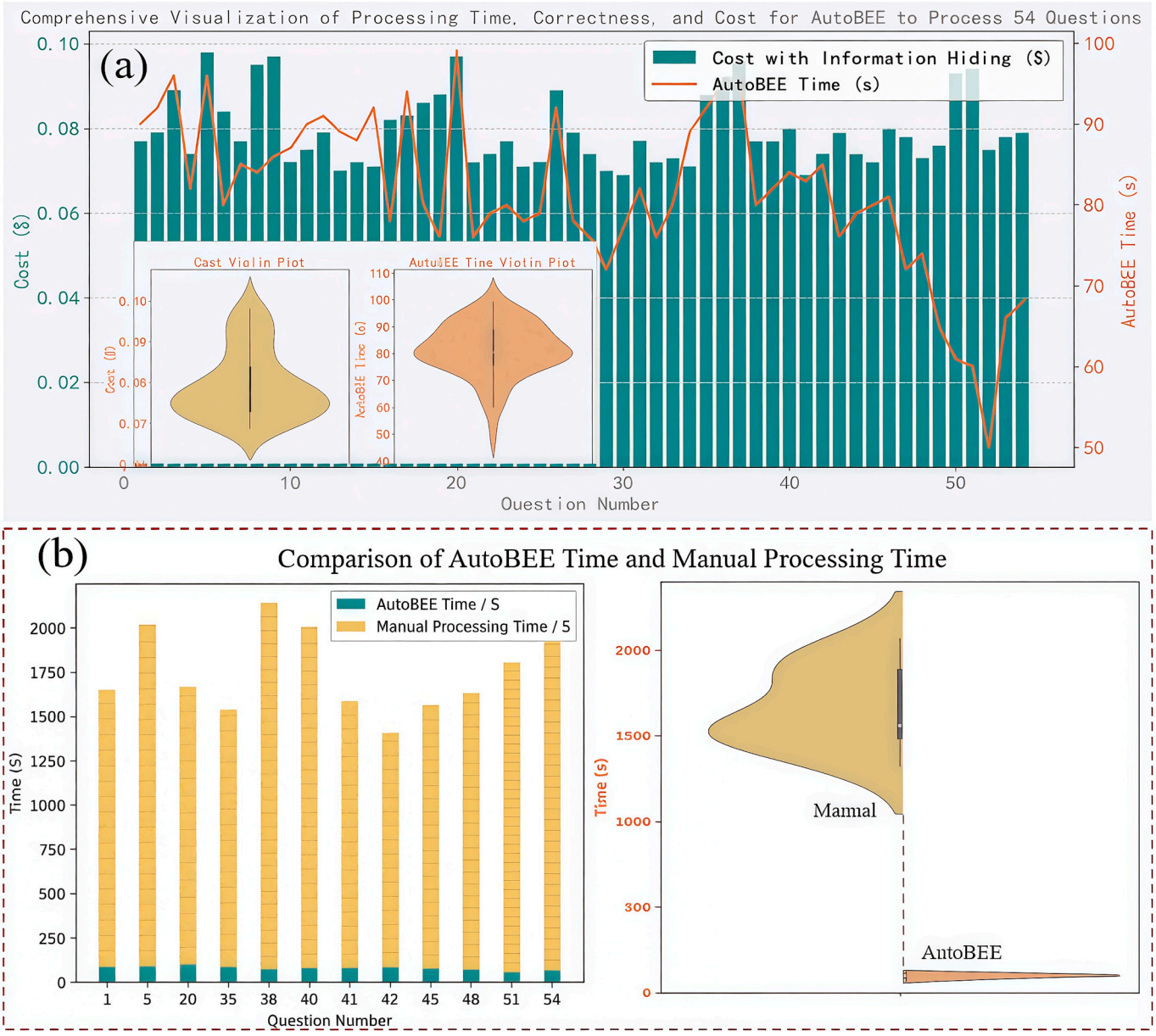


Fig. 7. Experimental Results of Time and Cost for 54 Application Cases.

(3) Rationality.

In the evaluation of rationality and content richness, for the 12 problems processed by both manual methods and AutoBEE, as shown in Fig. 8, AutoBEE demonstrated outstanding advantages. In terms of rationality, the logical coherence and method applicability of the reports generated by AutoBEE were comparable to those of the manual reports, and even superior in some cases, ensuring the rigor of the analysis process.

It is particularly worth mentioning that, in terms of content richness, the reports generated by AutoBEE were significantly better than those produced manually. This was due to its powerful natural language processing capabilities, which enabled it to quickly and comprehensively integrate multi-source data, deeply explore information related to building energy consumption and environmental parameters, and transform complex simulation data into reports with clear logic and substantial content. The reports not only fully presented various key indicators but also performed in-depth correlation analysis and trend prediction, providing more abundant and comprehensive information for building performance research and strongly supporting relevant decision-making and optimization work.

(4) Economy.

As shown in Fig. 7(a), the cost situation when AutoBEE used the information hiding strategy to process 54 problems can be clearly observed. The cost for adopting the information hiding strategy mostly ranged from 0.069 to 0.098($), and the overall cost was kept within a reasonable range. This demonstrated the effectiveness of the strategy in controlling costs.

Fig. 9 presents the comparison results of the information hiding strategy. When no information hiding was applied, the majority of tasks (approximately 92.6 %) failed, either falling into a loop or causing the program to terminate. This was due to the fact that, when tackling complex problems like building performance analysis, agents need to process a large volume of task-related information in multiple stages, including semantic understanding, parameter optimization, and simulation execution. Without the information hiding strategy, excessive and redundant information interfered with the agents, making it difficult for them to accurately identify key information. As a result, task planning and execution were disrupted, preventing tasks from progressing as

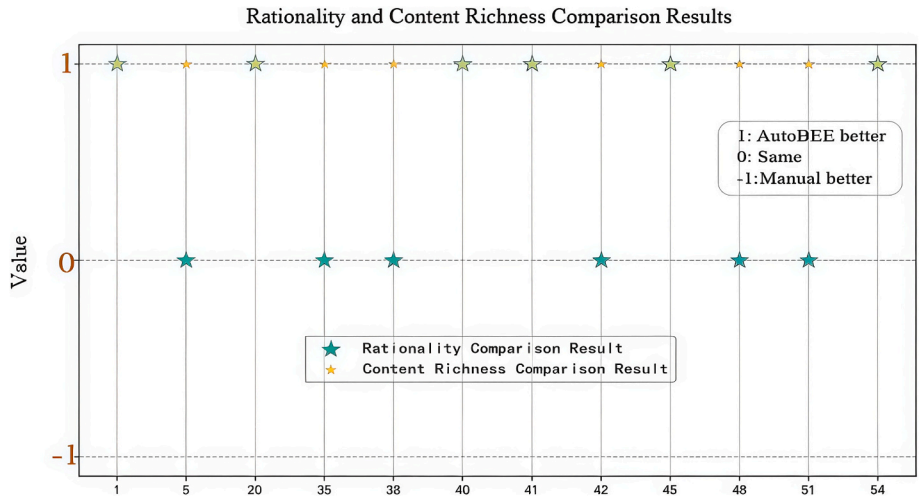


Fig. 8. Comparison of result quality between AutoBEE and manual processing.

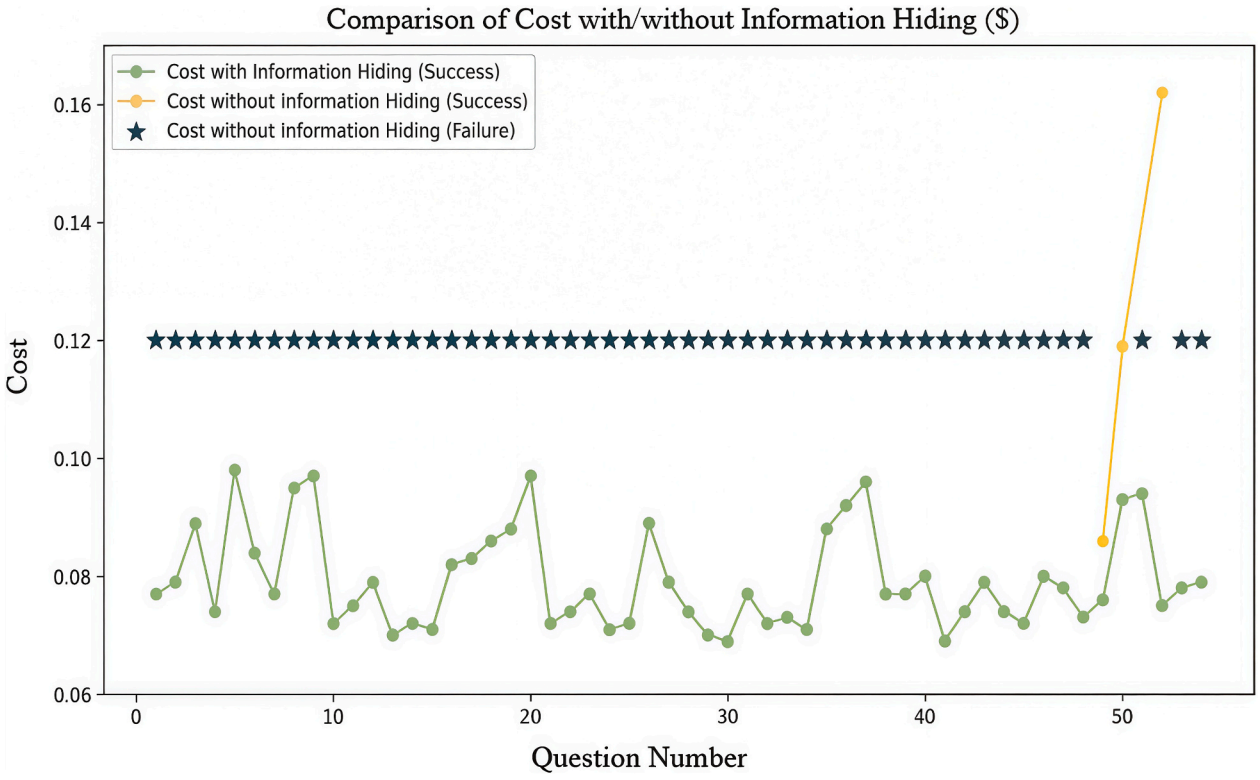


Fig. 9. Comparison of cost with/without Information Hiding.

intended. Only a few tasks (such as Problems 49, 50, and 52) were completed, but the cost increased dramatically. For instance, the cost of Problem 52 was 0.075($) with information hiding, but this figure increased to 0.162($) when no information hiding was applied. This stark contrast clearly highlights the importance and effectiveness of the information hiding strategy in improving task completion and controlling costs. By reducing token consumption, the strategy lowered operational costs and ensured both the economic viability and stability of the AutoBEE framework in handling building performance analysis tasks.

4. Discussion

To build a complete automated analysis framework for BECPA, AutoBEE, a multi-level system composed of teams and agents, was established. Through the improvement of individual agent capabilities and the enhancement of collaborative capabilities among agent groups, this system demonstrates High Efficiency, Accuracy, Rationality, Content Richness, and Economy when facing BECPA problems. The core technical mechanisms here include: the construction of a building performance analysis tool library, the establishment of a multi-level framework, the introduction of multi-level supervisory nodes, and the information hiding strategy. The following will discuss the application potential and limitations of AutoBEE.

4.1. Application potential of the framework

The design concept and technical architecture of the AutoBEE framework exhibit strong universality. It is not only applicable to energy

consumption simulation software, such as EnergyPlus, but also theoretically adaptable to any callable professional software or encapsulated specific function programs, such as computational fluid dynamics (CFD) software and building thermal environment simulation software. This versatility is attributed to the modular design and standardized interfaces of the framework, which encapsulate software call functions into pluggable tool modules. For example, in the case of CFD analysis, by linking the CFD software's call interface with the tool modules in the framework and optimizing the task planning and data processing logic of agents to align with the specific characteristics of CFD tasks, a fully automated process, from user instructions to the analysis of CFD simulation results, can be achieved. This universality significantly broadens the application scope of the framework, offering new perspectives for the intelligent resolution of complex engineering problems across various fields. Furthermore, it strongly contributes to the advancement of LLM towards becoming cross-disciplinary intelligent assistants.

4.2. Limitations and challenges

Although the AutoBEE framework has demonstrated excellent performance in experiments, there are still areas that require improvement. First, the coverage of experimental verification scenarios is somewhat limited. While it includes 54 typical building performance analysis scenarios, more complex coupling scenarios encountered in real world engineering remain unaddressed. Examples include joint analysis of dynamic energy consumption and indoor air quality in buildings across multiple climate zones, as well as the collaborative optimization of energy systems in large building complexes. These scenarios involve cross-disciplinary knowledge and strong multi-parameter coupling relationships, placing higher demands on the task decomposition, knowledge reasoning, and collaborative capabilities of the framework. The problems encountered during the experimental verification process mainly fall into three categories. First, process interruptions arise: while complex tasks require decomposition into multiple execution steps, the planning and supervision nodes fail to map out a clear path for task completion, leading to disruptions. Second, tool-calling errors occur: complex problems involve the collaborative use of multiple tools, yet the agent struggles to accurately match the required tools or provide the corresponding parameters, resulting in calling failures. Third, ambiguous answers appear in reports: as the number of overall task steps increases, the agent's sensitivity to the user's initial question diminishes, which in turn degrades the quality of the analysis report and leads to unclear conclusions. Therefore, there is an urgent need to improve the framework's adaptability to complex scenarios.

Second, there is still room for expansion in the functional division of agent teams within the framework. Currently, only basic teams such as planning, supervision, modification, simulation, and reporting have been established. This setup is insufficient to meet the needs of certain specialized tasks, For example, the task of converting natural language into initial IDF files in previous studies could be incorporated into the team. In the future, it will be necessary to further subdivide the roles of agents and introduce additional professional teams to build a more comprehensive agent ecosystem.

Although the AutoBEE framework still has limitations, it is undeniable that it provides a framework for constructing multi-level agent collaborative analysis of building energy consumption, and offers solutions for both individual agent capabilities and communication capabilities, laying a foundation for further development.

4.3. Future researches

Based on the above analysis, future research can be carried out in the following directions. First, it is necessary to further explore adaptability to complex scenarios. To address the multi-scenario and high-coupling problems in actual engineering, it is essential to expand the agent tool library and enhance team collaboration capabilities. Second, optimizing the architecture of agent teams is required. On the existing basis, new specialized teams should be added to adapt to problems in more dimensions and expand the scope of solvable issues, such as incorporating previous work on generating IDF files from natural language. Through these improvements and expansions, the AutoBEE framework is expected to evolve into a more powerful and versatile intelligent analysis platform.

5. Conclusion

This study developed AutoBEE, an automated analysis framework for building energy consumption and environmental parameters analysis. Built upon a hierarchical multi-agent system integrated with large language models, this framework focused on enhancing the efficiency of individual agent and productive collaboration among agent groups, achieving fully automated and unmanned building performance analysis. Specifically, this study has realized the automated process from users' natural language input to the output of building-related reports through the following three approaches:

(1) constructing a complete agent tool library covering building performance analysis;

(2) establishing a multi-level work network from teams to agents;

(3) developing a lightweight communication mechanism and a dynamic path selection framework.

In 54 experimental scenarios covering the full spectrum of building performance analysis, AutoBEE consistently outperformed traditional manual methods across five key dimensions: efficiency, accuracy, logical soundness, content richness, and cost effectiveness. It achieved a 10- to 20-fold improvement in task processing speed, delivered analytical reports with 100 % accuracy, and produced outputs that were more comprehensive and logically coherent, all while maintaining significantly lower operational costs.

In conclusion, AutoBEE provided an innovative, efficient, and reliable intelligent solution for building performance research, demonstrating strong potential to advance the analysis of energy consumption and environmental parameters toward greater automation and intelligence. Future research will aim to extend AutoBEE applicability to more complex and integrated scenarios, and further optimize its multi-agent architecture for interdisciplinary tasks.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used GPT4o in order to improve language. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

CRediT authorship contribution statement

Yani Quan: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Tong Xiao:** Validation, Methodology, Data curation. **Jiefan Gu:** Visualization, Project administration, Data curation. **Peng Xu:** Writing – review & editing, Supervision, Funding acquisition.

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.

Acknowledgement

This research is funded by National Natural Science Foundation of China (No.52161135202).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.enbuild.2025.116516.

Data availability

Data will be made available on request.

References

[1] Z. Wang, Y. Hong, L. Huang, et al., A comprehensive review and future research directions of ensemble learning models for predicting building energy consumption, Energ. Buildings 335 (2025) 115589, https://doi.org/10.1016/j.enbuild.2025.115589.

[2] A. Al-Shargabi, A. Almhafdy, D. Ibrahim, et al., Buildings' energy consumption prediction models based on buildings' characteristics: Research trends, taxonomy, and performance measures, Journal of Building Engineering. 54 (2022) 104577, https://doi.org/10.1016/j.jobe.2022.104577.

[3] N. Bucarelli, N. El-Gohary, Sensor deployment configurations for building energy consumption prediction, Energ. Buildings 308 (2024) 113888, https://doi.org/10.1016/j.enbuild.2024.113888D.

[4] J. Ji, H. Yu, X. Wang, X. Xu, Machine learning application in building energy consumption prediction: A comprehensive review, Journal of Building Engineering. 104 (2025) 112295, https://doi.org/10.1016/j.jobe.2025.112295.

[5] L. Zhang, Z. Chen, V. Ford, Advancing building energy modeling with large language models: Exploration and case studies, Energy & Buildings. 323 (2024) 114788.

[6] V.F. Mendes, A.S. Cruz, A.P. Gomes, J.C. Mendes, A systematic review of methods for evaluating the thermal performance of buildings through energy simulations, Renew. Sustain. Energy Rev. 189 (2024) 113875, https://doi.org/10.1016/j.rser.2023.113875.

[7] S. Norouzi, N. Chittoor, K. Grewal, et al., Geometric data in urban building energy modeling: Current practices and the case for automation, Journal of Building Engineering. 97 (2024) 110836, https://doi.org/10.1016/j.jobe.2024.110836.

[8] Y. Lu, A. Aleta, C. Du, et al., LLMs and generative agent - based models for complex systems research, Phys. Life Rev. 51 (2024) 283–293, https://doi.org/10.1016/j.plrev.2024.10.013.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Proces. Syst. 30 (2017) 6000–6010, https://doi.org/10.5555/3295222.3295349.

[10] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, A survey on large language model based autonomous agents, Front. Comp. Sci. 18 (6) (2024) 186345, https://doi.org/10.1007/s11704-024-40231-1.

[11] T. Xiao, P. Xu, Exploring automated energy optimization with unstructured building data: A multi-agent based framework leveraging large language models, Energ. Buildings 322 (2024) 114691, https://doi.org/10.1016/j.enbuild.2024.114691.

[12] S. Cheng, Z. Zhuang, Y. Xu, F. Yang, C. Zhang, X. Qin, Q. Zhang, Call Me When Necessary: LLMs can Efficiently and Faithfully Reason over Structured Environments, Findings of the Association for Computational Linguistics: ACL 2024 (2024) 4275–4295, https://doi.org/10.18653/v1/2024.findings.

[13] J. Lu, X. Tian, C. Zhang, Y. Zhao, J. Zhang, W. Zhang, C. Feng, J. He, J. Wang, F. He, Evaluation of large language models (LLMs) on the mastery of knowledge and skills in the heating, ventilation and air conditioning (HVAC) industry, Energy Built Environ. (2024), https://doi.org/10.1016/j.enbenv.2024.03.010.

[14] L. Chen, A. Darko, F. Zhang, A.P.C. Chan, Q. Yang, Can large language models replace human experts? Effectiveness and limitations in building energy retrofit challenges assessment, Build. Environ. 276 (2025) 112891.

[15] L. Song, C. Zhang, L. Zhao, J. Bian, Pre-Trained Large Language Models for Industrial Control. (2023), https://doi.org/10.48550/arXiv.2308.03028.

[16] Y. Li, M. Ji, J. Chen, X. Wei, X. Gu, J. Tang, A large language model - based building operation and maintenance information query, Energ. Buildings 334 (2025) 115515, https://doi.org/10.1016/j.enbuild.2025.115515.

[17] Z. Zheng, S. Marié, E. Farazdaghi, E. Yahia, K. Makhouli, T. Lagarde, R. El Meouche, F. Abobsa, Mastering building management systems data points tagging with minimal examples: unveiling the power of large language models, Energ. Buildings 328 (2025) 115173, https://doi.org/10.1016/j.enbuild.2024.115173.

[18] L. Zhang, Z. Chen, Large language model - based interpretable machine learning control in building energy systems, Energ. Buildings 313 (2024) 114278.

[19] S. Choi, S. Yoon, GPT - based data - driven urban building energy modeling (GPT-UBEM): Concept, methodology, and case studies, Energ. Buildings 325 (2024) 115042.

[20] J. Zheng, M. Fischer, Dynamic prompt-based virtual assistant framework for BIM information search, Autom. Constr. 155 (2023) 105067, https://doi.org/10.1016/j.autcon.2023.105067.

[21] M. Arslan, L. Mahdjoubi, S. Munawar, Driving sustainable energy transitions with a multi - source RAG - LLM system, Energ. Buildings 324 (2024) 114827, https://doi.org/10.1016/j.enbuild.2024.114827.

[22] J. Zhang, C. Zhang, J. Lu, Y. Zhao, Domain-specific large language models for fault diagnosis of heating, ventilation, and air conditioning systems by labeled-data supervised fine - tuning, Appl. Energy 377 (A) (2025) 124378.

[23] Y. Zhang, D. Wang, G. Wang, P. Xu, Y. Zhu, Data-driven building load prediction and large language models: Comprehensive overview, Energ. Buildings 326 (2025) 115001, https://doi.org/10.1016/j.enbuild.2024.115001.

[24] G. Jiang, Z. Ma, L. Zhang, J. Chen, EPlus-LLM: A large language model-based computing platform for automated building energy modeling, Appl. Energy 367 (2024) 123431, https://doi.org/10.1016/j.apenergy.2024.123431.

[25] L. Zhang, V. Ford, Z. Chen, J. Chen, Automatic building energy model development and debugging using large language models agentic workflow, Energ. Buildings 327 (2025) 115116, https://doi.org/10.1016/j.enbuild.2024.115116.

[26] L. Zhang, X. Fu, Y. Li, J. Chen, Large language model-based agent Schema and library for automated building energy analysis and modeling, Autom. Constr. 176 (2025) 106244, https://doi.org/10.1016/j.autcon.2025.106244.

[27] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, J. Wen, A survey on large language model based autonomous agents, Front. Comp. Sci. 18 (2024) 186345.

[28] Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, … Wang H. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv: 2312.10997. 2024; DOI: 10.48550/arXiv.2312.10997.

[29] LangChain, Multi-agent system examples, https://langchain-ai.github.io/langgraph/agents/multi-agent/, 2025.

[30] J. Wang, Y. Li, Y. Hong, Y. Tang, Integrated adaptive communication in multi - agent systems: Dynamic topology, frequency, and content optimization for efficient collaboration, Neurocomputing 617 (2025) 129068, https://doi.org/10.1016/j.neucom.2024.129068.

[31] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Zhou D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv preprint arXiv: 2201.12303. 2023; DOI: 10.48550/arXiv.2201.11903.

[32] J. Wang, Z. Duan, Agent AI with LangGraph: A Modular Framework for Enhancing Machine Translation Using Large Language Models, arXiv preprint arXiv: 2412.03801, https://arxiv.org/abs/2412.03801, 2024.

[33] LangChain, LangSmith, https://www.langchain.com/langsmith, 2025.