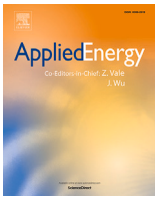
Applied Energy 415 (2026) 127976

Contents lists available at ScienceDirect

Applied Energy

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



Adaptive home energy management to self-motivated user preferences via iterative LLM-augmented reinforcement learning☆

Xiao Du[a], Fengji Luo[b], Juntao Hu[a], Wei Zhou[a], Junhao Wen[a,*]

[a] School of Big Data & Software Engineering, Chongqing University, Shapingba, Chongqing, 401331, Chongqing, China
[b] School of Civil Engineering, University of Sydney, Camperdown, Sydney, 2050, NSW, Australia

HIGHLIGHTS

- LLM-augmented framework adapts HEMS to natural-language user preferences.
- Three-agent loop reshapes DRL rewards and states via offline refinement.
- Deployed real-time policy operates independently without LLM inference.
- Reduces failure rate from 34.3% to 5.4% and ensures high rule compliance.
- Robust across multiple DRL algorithms, LLM backbones, and languages.

ARTICLE INFO

Keywords:
User preferences
Home energy management
Reinforcement learning
Large language model
Multi-agent system

ABSTRACT

Home energy management systems (HEMS) must balance competing objectives—electricity cost, thermal comfort, and carbon emissions—according to user preferences that are personalized, evolving, and often expressed in natural language. Conventional deep reinforcement learning (DRL) methods cannot directly interpret such preferences, while existing reinforcement learning (RL) + LLM integrations either invoke large language models (LLMs) at every control timestep or rely on one-shot preference parsing without feedback. This paper proposes LA-UPAHEM, an iterative LLM-augmented framework in which three specialized agents collaborate in a closed loop: a Code Generation Agent translates preferences into reward and state modification functions, a Result Analysis Agent diagnoses policy–preference misalignment from evaluation metrics, and an Optimal Performance Search Agent identifies the best-performing policy as the warm-start for the next iteration. LLMs are invoked only during this offline refinement phase; the deployed policy operates independently without LLM inference. Experiments on real residential energy datasets show that LA-UPAHEM outperforms classical DRL and existing RL + LLM baselines in both macro-level key performance indicator (KPI) alignment (cost, comfort, emissions) and micro-level rule compliance, achieving a Weighted Improvement Ratio (WIR) of 0.13 and a Rule Compliance Rate (RCR) of 0.84, while reducing the failure rate from 34.3% (static parsing) to 5.4%. The framework is robust to environmental noise, evaluation weight perturbation, and preference paraphrasing, and generalizes across multiple DRL algorithms, LLM backbones, and preference languages.

1. Introduction

1.1. Background

Residential energy consumption is growing in both scale and complexity, with household energy use rising by over 17% in several developed countries by 2021 [1] and residential buildings becoming the largest energy-consuming sector in the United States [2]. The proliferation of air conditioners (ACs), battery energy storage systems (BESS), and rooftop photovoltaic (PV) systems creates opportunities for efficiency but also introduces operational challenges: intermittent renewable generation, volatile electricity prices, and comfort needs that vary by time and lifestyle. Home energy management has consequently

☆ This work was supported by the Coal-Major Project (Grant No. 2025ZD1700800) and the Australian Research Council (Grant No. DP220103881).
* Corresponding author.
Email addresses: duxiao@cqu.edu.cn (X. Du), fengji.luo@sydney.edu.au (F. Luo), hujuntao@stu.cqu.edu.cn (J. Hu), zhouwei@cqu.edu.cn (W. Zhou), jhwen@cqu.edu.cn (J. Wen).

https://doi.org/10.1016/j.apenergy.2026.127976
Received 9 December 2025; Received in revised form 21 March 2026; Accepted 21 April 2026
Available online 24 April 2026

evolved into dynamic, multi-objective optimization balancing cost, comfort, and environmental impact.

Central to this optimization is understanding user preferences. A resident's notion of "optimal" energy usage reflects personal trade-offs among cost, comfort, and emissions—trade-offs that are subjective, dynamic, and shift with weather, routines, and personal circumstances. A Home Energy Management System (HEMS) that cannot adapt to these evolving preferences risks reduced user satisfaction and missed optimization opportunities.

1.2. Motivation

Traditional HEMS based on Rule-Based Control (RBC) or Model Predictive Control (MPC) struggle with the uncertainties in weather and user demands. Deep Reinforcement Learning (DRL) offers a promising alternative, combining adaptive decision-making with nonlinear modeling to learn control strategies directly from interaction data without explicit dynamic models. However, conventional DRL-based HEMS face two fundamental gaps:

1. Most DRL-HEMS are designed around static reward functions and fixed state spaces, defined by engineers before deployment. Translating a statement like "Only run the AC when I'm home and during the night" into precise DRL objectives requires domain knowledge and coding effort, making it inaccessible to everyday users. Even when preferences are known, encoding them into mathematical form is challenging because they are qualitative, context-dependent, and high-dimensional. Manual translation from natural language to DRL objectives is slow and error-prone, leaving systems unable to respond quickly to preference changes.
2. Existing "user-aware" HEMS approaches often rely on long-term statistical models, behavior tracking, or manual tuning to adjust control strategies. These methods suffer from delayed adaptation—they require significant data accumulation before adjusting—and cannot instantly respond to new or spontaneous user requests. Moreover, while Large Language Models (LLMs) excel at interpreting nuanced natural language and linking it to structured actions, their potential to serve as a continuous, adaptive bridge between user expression and DRL control logic remains underexplored in the HEMS context.

1.3. Contribution and scope

To close these gaps, we propose LA-UPAHEM (LLM-Augmented User-Preference-Adaptive HEMS), a framework that leverages LLM agents to interpret evolving user preferences expressed in natural language, translate them into modifications of DRL's state representation and reward structure, and iteratively refine control strategies while decoupling LLM reasoning from real-time device control to preserve system efficiency. By embedding natural language interpretation into the control optimization loop, LA-UPAHEM enables HEMS to maintain alignment with user priorities—even as those priorities change—without requiring residents to engage with the technical details of DRL. Our main contributions are:

1. We formulate the problem of natural language preference-driven HEMS, for the first time mapping user instructions expressed in natural language into DRL objectives via adaptive state augmentation and reward reshaping. Unlike prior user-aware HEMS that rely on predefined weight vectors or historical behavior inference, this formulation enables the direct translation of qualitative, context-dependent preferences into executable DRL components without manual reward engineering.
2. We design LA-UPAHEM, a multi-agent iterative LLM framework with three specialized agents (generation, analysis, and search). Unlike existing RL + LLM integrations that either invoke LLMs at every control timestep or rely on static one-shot preference parsing, LA-UPAHEM progressively refines reward and state functions through closed-loop feedback while decoupling LLM reasoning from real-time control—the deployed policy operates independently without LLM inference.
3. We conduct comprehensive empirical validation using real residential energy and weather datasets across eight preferences covering three categories and a preference-agnostic baseline. LA-UPAHEM achieves a WIR of 0.13 averaged over macro-level preferences and an RCR of 0.84 under behavioral-rule preferences, reduces the failure rate from 34.3% to 5.4%, and generalizes across multiple DRL algorithms, LLM backbones, and languages.

2. Related work

2.1. DRL in HEMS

DRL has emerged as a powerful paradigm for HEMS, offering data-driven solutions under uncertain and dynamic environments [3]. At the household level, DRL frameworks adapt to time-of-use tariffs and real-time battery operation [4], while robust approaches incorporate uncertainties in photovoltaic generation and user behavior [5]. At the community scale, multi-agent DRL enables large-scale coordination among residences [6], with extensions for safety constraints [7], explainability [8], and privacy [9]. Despite this maturity, most DRL-based HEMS assume static objectives. The challenge of modeling evolving user preferences and integrating language models into decision-making remains largely unexplored.

2.2. Modeling user preferences in HEMS

Modeling user preferences in HEMS remains challenging. Traditional approaches encode comfort as static penalty weights [10], which overlook the dynamic nature of household priorities and risk user disengagement when once-tuned strategies become suboptimal [11,12]. Multi-objective reinforcement learning (MORL) addresses this partially by maintaining separate value estimates per objective, enabling faster re-customization [13]. Similarly, satisfaction-based multi-agent approaches [14] and hierarchical control schemes [15] balance cost and comfort more flexibly. Recent work also introduces fuzzy logic to dynamically adjust comfort set-points [16]. Yet these methods still assume structured, bounded preferences and struggle with abrupt changes in occupant priorities.

An alternative direction infers preferences from historical behavior. [17] proposed an uncertainty-aware HEMS that learns appliance usage patterns via non-intrusive load monitoring (NILM) with Bayesian neural networks. However, such models adapt gradually and break under sudden deviations from routine (e.g., new work-from-home schedules). More broadly, rare events are usually missing from training data and "not feasible to capture" [18], leaving data-driven HEMS brittle against rapid preference shifts [19]. In summary, most preference-aware HEMS remain insufficiently adaptive: static parameters or history-based models fail to capture real-time variability when preferences change.

2.3. Integrating LLM with DRL

LLMs exhibit strong few-shot generalization [20] and can interpret natural language preferences [21], while DRL learns through interaction but suffers from high sample complexity [22]. This suggests natural complementarity, yet LLMs are costly and unsuited for real-time control. Existing RL + LLM integrations can be grouped by their involvement in real-time control [23]: (i) directly employing LLMs to output control actions [24–27]; (ii) using LLMs to provide auxiliary decision information such as skill priors or action filters [28–30]; and (iii) leveraging LLMs to generate static training guidance, such as reward functions [31,32], regularization functions [33], intrinsic motivation signals [34], or state representations [35]. Table 1 summarizes these paradigms.

However, the first two categories face practical challenges: LLM inference is computationally expensive for real-time coupling [36], and LLMs remain prone to hallucinations in planning tasks [23,37].

Table 1
Methodological comparison of RL + LLM integration paradigms in the literature.

Paradigm	Representative works	LLM role	When invoked	Iterative
I. Action gen.	[24–27]	Policy / actor	Every timestep	✗
II. Auxiliary info.	[28–30]	Filter / prior	Per episode / timestep	✗
III. Static guidance	[31–35]	Guidance designer	Once	✗
LA-UPAHEM	This work	Reward + state designer	Per iteration	✓

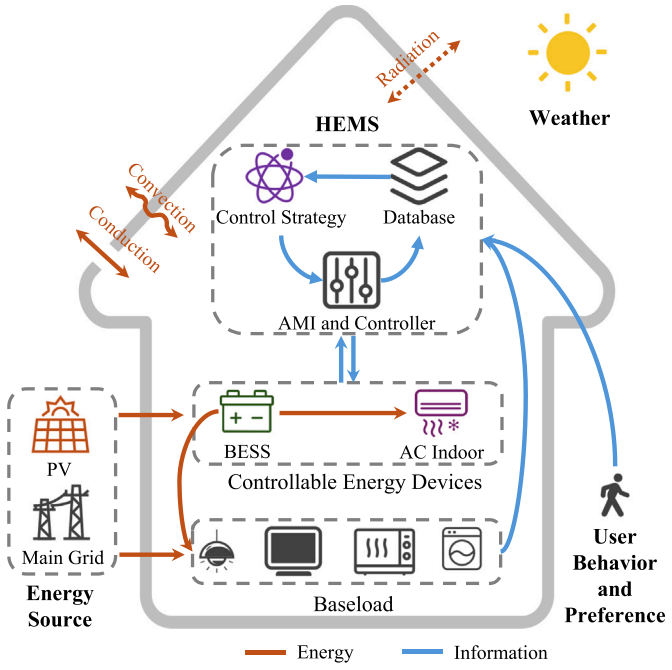


Fig. 1. System architecture of the considered HEMS scenario.

This work builds upon the third category but introduces two key innovations. First, rather than static reward generation without verification, we propose an iterative closed-loop mechanism where the Result Analysis Agent evaluates policy performance and provides corrective feedback, enabling progressive refinement of reward functions and state modification functions. Second, we systematically apply this paradigm to residential energy management, where natural language preferences are common yet unaddressed by existing LLM + DRL approaches. By decoupling LLM reasoning from real-time control, the proposed framework avoids the computational and robustness issues of the first two categories while achieving adaptive preference alignment.

3. Problem formulation

3.1. HEMS objectives and scenario

We consider a modern residential environment with high renewable energy penetration that can accommodate diverse and evolving user preferences. As illustrated in Fig. 1, the household is supplied by both the main power grid and rooftop photovoltaic (PV) generation, while the HEMS actively controls two Controllable Energy Devices (CEDs): a Battery Energy Storage System (BESS) and an Air Conditioner (AC). Other household appliances (e.g., televisions, microwave ovens) are treated as uncontrollable baseline load P_t^{BL} (kW). The system is modeled in a modular manner, allowing devices to be added or removed without compromising the generality of the control framework.

The HEMS operates in discrete time steps $t \in \{1, 2, \ldots, \tau\}$ with interval Δt (h), observing household states through the Advanced Metering Infrastructure (AMI) and issuing control decisions for the CEDs. The core task of residential energy management is to balance three user-centric performance dimensions: minimizing electricity expenditure $\mathcal{L}_{\mathrm{c}}$, maintaining indoor thermal comfort $\mathcal{L}_{\mathrm{t}}$, and reducing carbon emissions $\mathcal{L}_{\mathrm{e}}$. These objectives are formalized through the following Key Performance Indicators (KPIs):

$$\mathcal{L}_{\mathrm{c}} = \frac{1}{\tau}\sum_{t=1}^{\tau} \kappa_t P_t^{\mathrm{G}} \Delta t, \quad \text{(Electricity Cost)} \tag{1a}$$

$$\mathcal{L}_{\mathrm{t}} = \frac{1}{\tau}\sum_{t=1}^{\tau} \mathbb{1}\left(|T_t^{\mathrm{in}} - T_t^{\mathrm{set}}| > T^{\mathrm{band}}\right), \quad \text{(Thermal Discomfort)} \tag{1b}$$

$$\mathcal{L}_{\mathrm{e}} = \frac{1}{\tau}\sum_{t=1}^{\tau} \mu_t P_t^{\mathrm{G}} \Delta t, \quad \text{(Carbon Emissions)} \tag{1c}$$

Here, κ_t (\$/kWh) and μ_t (kg CO_2/kWh) denote the time-varying electricity price and grid carbon intensity, P_t^{G} (kW) is the power exchanged with the grid (positive for import), T_t^{in} (°C) and T_t^{set} (°C) are the actual and target indoor temperatures, and T^{band} (°C) is the acceptable deviation band. While these KPIs capture essential trade-offs in residential energy management, real users often express preferences that go beyond simple KPI weighting, such as conditional device usage rules or context-dependent priorities.

3.2. MDP formulation

We formalize the HEMS control problem as a Markov Decision Process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. The state space $\mathcal{S}$ comprises environmental variables (e.g., outdoor temperature T_t^{out}, solar irradiance I_t), device states (e.g., indoor temperature T_t^{in}, BESS state of charge soc_t), contextual information (e.g., hour-of-day hour_t, occupancy indicator occ_t), and market signals (electricity price κ_t, carbon intensity μ_t). The action space $\mathcal{A}$ specifies operating modes and power levels for CEDs (e.g., BESS charging/discharging power P_t^{b}, AC operating mode M_t^{a}). The transition dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$ are governed by physical models and stochastic processes, as detailed in Section 4. The discount factor is set to $\gamma = 0.99$ to encourage long-horizon planning.

The reward function $\mathcal{R}(s_t, a_t)$ quantifies the desirability of state-action pairs. In traditional DRL-based HEMS, rewards are manually designed based on the KPIs in Eq. (1), and the overall reward is computed as a weighted sum $r_t = \alpha_{\mathrm{c}}\mathcal{R}_{\mathrm{c}} + \alpha_{\mathrm{t}}\mathcal{R}_{\mathrm{t}} + \alpha_{\mathrm{e}}\mathcal{R}_{\mathrm{e}}$ with manually tuned weights. The objective is to learn an optimal policy $\pi^\star(a_t|s_t)$ that maximizes the expected cumulative reward:

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{t=1}^{\tau} \gamma^t r_t\right]. \tag{2}$$

However, this reward structure is restricted to linear combinations of predefined components and cannot capture natural language preferences such as conditional device usage rules.

3.3. Preference-driven MDP

To overcome these limitations, we extend the standard MDP to accommodate natural language user preferences **P**. Importantly, the resulting problem remains a conventional numerical RL problem, not a multi-modal one: natural language is not processed jointly with numerical states during policy execution. Rather, LLM agents serve as an offline preprocessor that translates natural language preferences into numerical reward functions and state modification functions before DRL training begins. Once these components are generated, the MDP is purely numerical—the agent observes numerical states, takes continuous actions, and receives scalar rewards without any language modality. LLMs are invoked only during the iterative refinement phase and are entirely absent from the deployed real-time control loop.

Specifically, given a preference statement $\mathbf{P}$, task-specific context $\mathbf{C}$, and prompt templates $\mathbf{T}$, we seek a mapping:

$$\Phi : (\mathbf{P}, \mathbf{C}, \mathbf{T}) \to (\mathcal{M}_{\mathbf{P}}, \mathcal{R}_{\mathbf{P}}), \tag{3}$$

where $\mathcal{M}_{\mathbf{P}} : \mathcal{S} \to \tilde{\mathcal{S}}$ is a *state modification function* that augments the state space with preference-relevant features, and $\mathcal{R}_{\mathbf{P}} : \tilde{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}$ is a preference-driven reward function defined over the augmented state space. The task-specific context $\mathbf{C}$ encapsulates household-specific information (device configuration, environment parameters) and must be customized per scenario, whereas the prompt templates $\mathbf{T}$ are standardized instructions that remain fixed across deployments. Since $\mathbf{C}$ and $\mathbf{T}$ rarely change once established, users only need to update $\mathbf{P}$ to express new preferences. The construction of $\mathbf{C}$ and $\mathbf{T}$ is detailed in Section 5.

Note that $(\mathcal{M}_{\mathbf{P}}, \mathcal{R}_{\mathbf{P}})$ are generated as a coordinated pair: the state modification function $\mathcal{M}_{\mathbf{P}}$ provides the features that $\mathcal{R}_{\mathbf{P}}$ depends on. The extended MDP becomes $\langle \tilde{\mathcal{S}}, \mathcal{A}, \mathcal{P}, \mathcal{R}_{\mathbf{P}}, \gamma \rangle$, and the policy is optimized as:

$$\pi^{\star}_{\mathbf{P}} = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{\tau} \gamma^t \mathcal{R}_{\mathbf{P}}(\tilde{s}_t, a_t)\right], \quad \text{where } \tilde{s}_t = \mathcal{M}_{\mathbf{P}}(s_t). \tag{4}$$

Unlike traditional formulations, $\mathcal{R}_{\mathbf{P}}$ is not restricted to linear combinations of predefined reward components. It can be any differentiable function tailored to the preference, including nonlinear terms, conditional penalties, or entirely new reward structures. For instance, the preference "*do not charge the battery when I am at home*" requires $\mathcal{M}_{\mathbf{P}}$ to augment the state with occupancy indicators, which $\mathcal{R}_{\mathbf{P}}$ then uses to impose conditional penalties on charging actions during occupied periods.

3.4. Preference categories

We classify user preferences into three categories, reflecting the primary trade-off dimensions identified in the HEMS literature: electricity cost versus thermal comfort [13,14,17] and carbon emissions awareness [18]. Macro-level preferences specify high-level KPI trade-offs (e.g., $\mathbf{P}_1$: "*sacrifice comfort for cost*"), typically implemented by adjusting reward weights or functional forms. Micro-level preferences impose fine-grained device operation rules (e.g., $\mathbf{P}_4$: "*no battery charging at home*"), requiring state augmentation and conditional penalties. Compositional preferences combine both (e.g., $\mathbf{P}_6$: "*tolerate cold to save cost, no AC below setpoint*"). These three categories are not exhaustive but cover the most common preference types reported in residential energy surveys; the framework's code-generation mechanism can accommodate preferences beyond these categories as long as they can be expressed in natural language. Table 2 summarizes the eight preferences used throughout this study.

4. System modeling

To capture the physical and stochastic characteristics of the residential environment, we develop models for the CEDs, rooftop PV generation, and the building's thermal dynamics. These components collectively define the environmental dynamics that govern the HEMS optimization and control process.

4.1. Energy device models

4.1.1. Battery energy storage system (BESS)

The BESS follows standard charge/discharge dynamics with stored energy E_t (kWh), rated capacity E_{rat} (kWh), and state of charge (SOC) $\text{soc}_t = E_t / E_{\text{rat}}$:

$$E_{t+1} = \overline{E}_{t+1}(1 - \text{SD} \cdot \Delta t), \tag{5a}$$

$$\overline{E}_{t+1} = \begin{cases} E_t + \eta^{\text{b}} P^{\text{b}}_t \Delta t, & M^{\text{b}}_t = 1, \\ E_t - \frac{P^{\text{b}}_t \Delta t}{\eta^{\text{b}}}, & M^{\text{b}}_t = -1, \\ E_t, & M^{\text{b}}_t = 0, \end{cases} \tag{5b}$$

Table 2
Categories of user preferences with illustrative examples. $\mathbf{P}_1$–$\mathbf{P}_8$ are used in the main experiments.

Symbol	Category	Natural language preference
$\mathbf{P}_1$	Macro	"*I am willing to sacrifice some room temperature comfort for electricity costs.*"
$\mathbf{P}_2$	Macro	"*I am an environmentalist, and I hope the carbon emissions of my home can be as low as possible.*"
$\mathbf{P}_3$	Macro	"*Please prioritize the comfort of my home.*"
$\mathbf{P}_4$	Micro	"*For safety reasons, I do not want the battery to be charged while I'm at home.*"
$\mathbf{P}_5$	Micro	"*I prefer to turn on the air conditioner only when I'm at home and during the night.*"
$\mathbf{P}_6$	Micro	"*I want the home battery to discharge only during peak electricity price hours to take advantage of time-of-use tariffs.*"
$\mathbf{P}_7$	Compositional	"*I can adapt to a colder room temperature, so there is no need to turn on the air conditioner when the room temperature is below the set temperature to save on electricity costs.*"
$\mathbf{P}_8$	Baseline	"*I have no particular preference.*"

$$P^{\text{b}}_t \in [0, P^{\text{b}}_{\text{sup}}], \tag{5c}$$

$$P^{\text{b}}_{\text{sup}} = \begin{cases} \min\{P^{\text{b}}_{\text{rat}}, \frac{E_{\text{rat}} - E_t}{\eta^{\text{b}} \Delta t}\}, & M^{\text{b}}_t = 1, \\ \min\{P^{\text{b}}_{\text{rat}}, \frac{\eta^{\text{b}} E_t}{\Delta t}\}, & M^{\text{b}}_t = -1, \\ 0, & M^{\text{b}}_t = 0, \end{cases} \tag{5d}$$

where $M^{\text{b}}_t \in \{-1, 0, 1\}$ denotes the operating mode (1: charging, −1: discharging, 0: standby), P^{b}_t (kW) and $P^{\text{b}}_{\text{rat}}$ (kW) are the actual and rated power, SD (h^{-1}) is the self-discharge rate, and η^{b} is the charge/discharge efficiency.

4.1.2. Air conditioner (AC)

The AC is modeled thermodynamically with heating/cooling power P^{heat}_t (kW) and coefficient of performance (COP) cop_t dependent on indoor temperature T^{in}_t (°C) and outdoor temperature T^{out}_t (°C):

$$P^{\text{heat}}_t = \text{cop}_t P^{\text{a}}_t M^{\text{a}}_t, \quad P^{\text{a}}_t \in [P^{\text{a}}_{\text{sta}}, P^{\text{a}}_{\text{rat}}], \tag{6a}$$

$$\text{cop}_t = \begin{cases} \overline{\text{cop}}_t, & 0 < \overline{\text{cop}}_t < \text{cop}_{\text{rat}}, \\ \text{cop}_{\text{rat}}, & \text{otherwise}, \end{cases} \tag{6b}$$

$$\overline{\text{cop}}_t = \begin{cases} \frac{\eta^{\text{a}} (T^{\text{in}}_t)_{\text{K}}}{T^{\text{in}}_t - T^{\text{out}}_t}, & M^{\text{a}}_t = 1, \\ \frac{\eta^{\text{a}} (T^{\text{in}}_t)_{\text{K}}}{T^{\text{out}}_t - T^{\text{in}}_t}, & M^{\text{a}}_t = -1, \\ 0, & M^{\text{a}}_t = 0, \end{cases} \tag{6c}$$

where $M^{\text{a}}_t \in \{-1, 0, 1\}$ is the operating mode (1: heating, −1: cooling, 0: standby), P^{a}_t (kW) is the electrical power consumption bounded by minimum $P^{\text{a}}_{\text{sta}}$ (kW) and rated $P^{\text{a}}_{\text{rat}}$ (kW) values, $(\cdot)_{\text{K}}$ denotes temperature conversion to Kelvin, cop_{rat} is the upper COP limit, and η^{a} is the thermodynamic efficiency factor.

4.1.3. Photovoltaic (PV) system

Energy generation P^{PV}_t (kW) depends on solar irradiance I_t (kW/m^2), panel area A^{pan} (m^2), conversion efficiency η^{PV}, and rated output per unit area $P^{\text{PV}}_{\text{rat}}$ (kW/m^2):

$$P^{\text{PV}}_t = \begin{cases} 0, & I_t < I_{\min}, \\ \min(\eta^{\text{PV}} I_t, P^{\text{PV}}_{\text{rat}}) A^{\text{pan}}, & \text{otherwise}, \end{cases} \tag{7}$$

where $I_{\min}$ (kW/m^2) is the minimum irradiance threshold for generation.

4.2. Building thermal model

The indoor temperature T^{in}_t (°C) evolves according to heat transfer from conduction Q^{con}_t (kWh), ventilation Q^{ven}_t (kWh), and solar radiation

Q_t^{solar} (kWh):

$$T_{t+1}^{\text{in}} = T_t^{\text{in}} + \frac{Q_t^{\text{con}} + Q_t^{\text{ven}} + Q_t^{\text{solar}} + P_t^{\text{heat}}\Delta t}{c^{\text{air}}\rho^{\text{air}}V^{\text{hou}}}, \tag{8a}$$

$$Q_t^{\text{con}} = \lambda A^{\text{suf}}(T_t^{\text{out}} - T_t^{\text{in}})\Delta t, \tag{8b}$$

$$Q_t^{\text{ven}} = \text{ACH} \cdot c^{\text{air}}\rho^{\text{air}}V^{\text{hou}}(T_t^{\text{out}} - T_t^{\text{in}})\Delta t, \tag{8c}$$

$$Q_t^{\text{solar}} = \text{SHGC} \cdot I_t A^{\text{fen}}\Delta t, \tag{8d}$$

where c^{air} (kWh/(kg· °C)) is the specific heat of air, ρ^{air} (kg/m^3) is air density, V^{hou} (m^3) is house volume, λ (kW/(m^2· °C)) is thermal conductivity, A^{suf} (m^2) is the envelope surface area, A^{fen} (m^2) is the window area, ACH (h^{-1}) is the air change rate, and SHGC is the solar heat gain coefficient.

4.3. Power balancing constraint

To satisfy household energy demand, the total power demand P_t^{D} and the power drawn from the main grid P_t^{G} are defined as:

$$P_t^{\text{D}} = P_t^{\text{a}} + P_t^{\text{b}} + P_t^{\text{BL}} - P_t^{\text{PV}}, \tag{9a}$$

$$P_t^{\text{G}} = \begin{cases} P_t^{\text{D}} & \text{if } P_t^{\text{D}} \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{9b}$$

where P_t^{BL} represents the uncontrollable baseline load from other household appliances. We assume no feed-in to the grid (i.e., excess PV generation is curtailed), which reflects regulatory constraints in certain residential markets.

4.4. Stochastic occupancy model

We consider distinct occupancy patterns for working days (work) and rest days (rest), each characterized by different probabilities of being away from home (p^{out}) and separate distribution parameters (μ, σ) for departure (t^{dep}) and arrival (t^{arr}) times. The occupancy indicator $\text{occ}_{d,t}$ for day d is modeled as follows:

$$\text{occ}_{d,t} = 1 - \mathbb{1}\left(t_d^{\text{dep}} \leq t < t_d^{\text{arr}}\right), \tag{10a}$$

$$t_d^{\{\text{dep,arr}\}} \sim \begin{cases} \mathcal{N}(\mu_{\{\text{work,rest}\}}^{\{\text{dep,arr}\}}, (\sigma_{\{\text{work,rest}\}}^{\{\text{dep,arr}\}})^2), & \text{if out}_d = 1, \\ \{0, 24\}, & \text{otherwise}, \end{cases} \tag{10b}$$

$$\text{out}_d \sim \text{Bernoulli}(p_{\{\text{work,rest}\}}^{\text{out}}). \tag{10c}$$

4.5. Exogenous variables from data

Modeling solar radiation density (I_t), baseline load (P_t^{BL}), and outdoor temperature (T_t^{out}) is challenging due to their inherent uncertainty and variability. Therefore, we leverage real-world historical datasets [38,39] to simulate their dynamics. The real-time electricity price (κ_t) and carbon emission intensity (μ_t) are obtained from the CityLearn v2 dataset [40]. All variables in the state vector $\mathbf{s}_t$—including the above exogenous signals, device operating states, and calendar features—are assumed directly measurable via Advanced Metering Infrastructure (AMI) or local sensors. Latent quantities such as the instantaneous AC coefficient of performance (COP), battery self-discharge losses, and heat transfer through the building envelope are computed internally by the simulation but are *not* included in $\mathbf{s}_t$.

4.6. Modeling assumptions

The models above operate under three key assumptions. First, excess PV generation is curtailed with no feed-in to the grid (Eq. 9), reflecting markets without net-metering. In jurisdictions that permit feed-in, the BESS would gain an export revenue stream, likely shifting learned strategies toward more aggressive battery cycling; the LA-UPAHEM framework can accommodate this by including export tariff data in the task-specific context $\mathbf{C}$ without architectural changes. Second, we consider a single household with two controllable devices (AC and BESS). Third, occupancy is modeled as a binary indicator (home/away) without multi-occupant disaggregation. Extending to multi-household coordination and richer occupancy models is left for future work.

Because exogenous variables are drawn from real-world traces rather than synthetic distributions, the simulation partially mitigates the gap between simulated and physical environments. Remaining discrepancies—such as simplified thermal dynamics or unmodeled disturbances (e.g., door openings, appliance heat gains)—can be further addressed through domain randomization over building parameters, periodic online policy fine-tuning with real operational data, or digital twin integration.

5. Methodology

To operationalize the preference-driven MDP formulation, we propose the LA-UPAHEM framework. LA-UPAHEM automatically translates natural language preferences into executable RL components and iteratively refines them based on training feedback. The overall architecture is illustrated in Fig. 2.

The framework comprises three stages detailed in the following subsections: task-specific context preparation (Section 5.1), LLM agent interaction via prompt templates (Section 5.2), and an adaptive iterative loop that couples DRL training with LLM-driven refinement (Section 5.4).

5.1. Task-specific context

The task-specific context $\mathbf{C}$ provides LLM agents with the necessary information to understand the HEMS deployment and generate appropriate $(\mathcal{M}_{\mathbf{P}}, \mathcal{R}_{\mathbf{P}})$ pairs. It consists of three elements:

Environment Description $\mathbf{c}_{\text{des}}$: An overview of the deployment environment, including the HEMS architecture, the set of CEDs, and natural language explanations of the action space $\mathcal{A}$ and state space $\mathcal{S}$. Each state variable is annotated with its physical meaning, data type, and valid range to facilitate accurate code generation.

General KPIs Description $\mathbf{c}_{\text{kpi}}$: High-level performance metrics corresponding to the minimization objectives $\mathcal{L}_{\text{c}}, \mathcal{L}_{\text{t}}, \mathcal{L}_{\text{e}}$ in Eq. (1). These provide semantic grounding for LLM agents to understand the quantitative meanings of "cost", "comfort", and "emissions".

Code Reference Templates ($\mathbf{c}_{\text{rref}}, \mathbf{c}_{\text{mref}}$): Executable Python code demonstrating the expected interface and design patterns for reward and state modification functions. The state modification template $\mathbf{c}_{\text{mref}}$ augments the base state s_t with derived features:

$$\mathcal{M}_{\text{c}}(s_t) = -|P_t^{\text{D}}| \cdot \kappa_t, \tag{11a}$$

$$\mathcal{M}_{\text{t}}(s_t) = \min\{0,\ T^{\text{band}} - |T_t^{\text{in}} - T_t^{\text{set}}|\}, \tag{11b}$$

$$\mathcal{M}_{\text{e}}(s_t) = -|P_t^{\text{D}}| \cdot \mu_t, \tag{11c}$$

where all symbols follow their definitions in Sections 3 and 4. $\mathcal{M}_{\text{c}}(s_t)$, $\mathcal{M}_{\text{t}}(s_t)$, and $\mathcal{M}_{\text{e}}(s_t)$ capture the electricity cost intensity, thermal discomfort degree, and carbon emission intensity, respectively. The reward function template $\mathbf{c}_{\text{rref}}$ then applies bounded transformations to these derived features:

$$\mathcal{R}_{\text{c}}(s_t, a_t) = \exp(\mathcal{M}_{\text{c}}(s_t)), \tag{12a}$$

$$\mathcal{R}_{\text{t}}(s_t, a_t) = \text{occ}_t \cdot \exp(\mathcal{M}_{\text{t}}(s_t)), \tag{12b}$$

$$\mathcal{R}_{\text{e}}(s_t, a_t) = \exp(\mathcal{M}_{\text{e}}(s_t)), \tag{12c}$$

The exponential form ensures bounded rewards in $(0, 1]$ and provides smooth gradients, which stabilize DRL training. The complete Python implementations of these reference templates are provided in Appendix D. These templates serve as reference examples that illustrate expected interfaces, design patterns, and how to incorporate conditional indicators and temporal encodings; the generated $\mathcal{M}_{\mathbf{P}}$ and $\mathcal{R}_{\mathbf{P}}$ (Eq. 3)

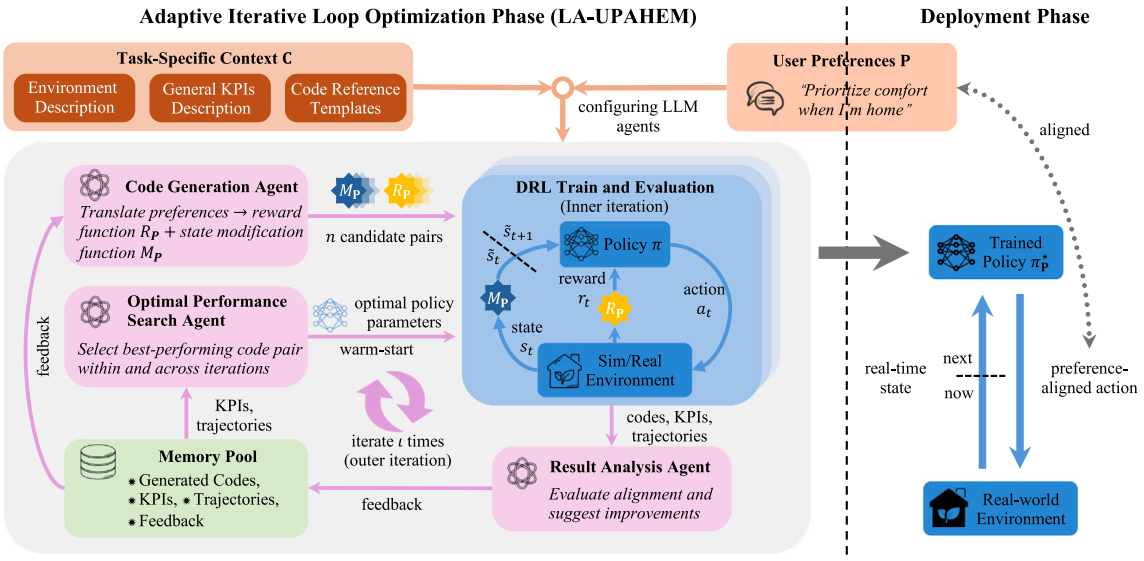


Fig. 2. Overall architecture of the proposed LA-UPAHEM framework.

are not restricted to the template structure and may introduce entirely new features or reward terms tailored to user preferences **P** (a concrete example is presented in the case study of Section 6.4).

Since **C** is determined by the household configuration and remains stable once established, users only need to provide **P** in natural language to express their preferences, significantly simplifying human-machine interaction.

5.2. LLM agents and prompt templates

The prompt templates $\mathbf{T} = (\mathbf{t}_{\text{code}}, \mathbf{t}_{\text{analysis}}, \mathbf{t}_{\text{search}})$ are standardized instructions that guide LLM agents to realize the mapping Φ in Eq. (3). Each template defines a distinct functional role, and the differentiation among these templates gives rise to three specialized agents. The complete prompt templates are provided in Appendix C.

The **Code Generation Agent**, driven by $\mathbf{t}_{\text{code}}$, translates user preferences into executable $(\mathcal{R}_\mathbf{P}, \mathcal{M}_\mathbf{P})$ pairs. Given the task-specific context **C**, user preferences **P**, and historical records from the memory pool, the agent generates n diverse candidate implementations:

$$\{(\mathcal{R}_\mathbf{P}^j, \mathcal{M}_\mathbf{P}^j)\}_{j=1}^n = \mathcal{F}_{\text{code}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{code}}, \mathcal{D}_{i-m:i-1}), \tag{13}$$

where i denotes the outer iteration index and $\mathcal{D}_{i-m:i-1}$ is a sliding window over the memory pool containing the most recent m iterations of generated code, evaluation results, and feedback (with $\mathcal{D}_{i-m:i-1} = \emptyset$ at $i = 1$). The agent parses **P** to identify preference types and key constraints, uses the code templates as structural references while modifying components based on preferences, and generates n diverse implementations exploring different reward structures. Generating multiple candidates enables both exploration of diverse strategies and exploitation through iterative refinement.

The **Result Analysis Agent**, driven by $\mathbf{t}_{\text{analysis}}$, evaluates training outcomes and generates targeted improvement feedback. After the inner loop trains and evaluates all n candidate strategies, this agent receives the evaluation results $\mathbf{e}_i$ and analyzes performance:

$$\mathbf{f}_i = \mathcal{F}_{\text{analysis}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{analysis}}, \mathbf{e}_i, \{(\mathcal{R}_\mathbf{P}^j, \mathcal{M}_\mathbf{P}^j)\}_{j=1}^n), \tag{14}$$

where $\mathbf{e}_i = \{\mathbf{k}_i, \mathbf{u}_i\}$ contains the final KPI values $\mathbf{k}_i = \{\mathcal{L}_\text{c}^j, \mathcal{L}_\text{t}^j, \mathcal{L}_\text{e}^j\}_{j=1}^n$ and downsampled control trajectories $\mathbf{u}_i$ from the test environment. The agent compares KPI values across candidates, analyzes control trajectories to detect preference violations, and identifies discrepancies between intended and actual behavior. We refer to the complete output of this agent as *feedback* $\mathbf{f}_i$: structured natural language that combines quantitative KPI analysis with specific code modification suggestions. For instance, a representative feedback excerpt might read: "*Sample 0 achieves the lowest electrical cost ($6.8) but has a high discomfort proportion (0.38). The reward function over-penalizes AC usage. Suggestion: reduce the cost weight from 0.5 to 0.3 and add an occupancy-conditional comfort term.*" This structured format provides actionable guidance that the Code Generation Agent can directly incorporate in the next iteration.

The **Optimal Performance Search Agent**, driven by $\mathbf{t}_{\text{search}}$, identifies the best-performing strategy based on preference alignment:

$$(i^\star, j^\star) = \begin{cases} \mathcal{F}_{\text{search}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{search}}, \mathbf{e}_i), & i < \iota, \\ \mathcal{F}_{\text{search}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{search}}, \{\mathbf{e}_i\}_{i=1}^{\iota}), & i = \iota, \end{cases} \tag{15}$$

where ι is the total number of outer iterations. For $i < \iota$, the agent selects the current best strategy $j^\star$ as the warm-start model for the next iteration, accelerating convergence by initializing from a well-performing policy. At $i = \iota$, it selects the globally optimal strategy $(i^\star, j^\star)$ across all iterations for final deployment.

Since the Code Generation Agent produces executable code that directly interfaces with the simulation environment, a multi-layer validation pipeline (interface constraints, syntactic validation, runtime testing, retry mechanism, and state space documentation) ensures the safety and correctness of all generated code. The complete description of these safety measures is provided in Appendix E.

5.3. DRL algorithm

LA-UPAHEM requires a DRL algorithm that can robustly optimize policies under dynamically evolving reward functions generated by LLM agents. We adopt the Soft Actor-Critic (SAC) algorithm [41], which incorporates entropy regularization into the objective function:

$$\max_{\pi_\theta \in \Pi} J(\pi_\theta) = \mathbb{E}_{\xi \sim \pi_\theta}\left[\sum_{t=1}^{\tau} \gamma^t \left(r_t + \alpha \mathcal{H}(\pi_\theta(\cdot|s_t))\right)\right], \tag{16}$$

where $\mathcal{H}(\pi_\theta(\cdot|s_t)) = -\mathbb{E}[\log \pi_\theta(a_t|s_t)]$ is the policy entropy and $\alpha > 0$ is the temperature parameter. SAC is particularly suitable for LA-UPAHEM for two reasons. First, the entropy regularization with automatic temperature adjustment encourages the policy to maintain appropriate stochasticity. This is important because LLM-generated reward functions may differ substantially across iterations, and exploratory behavior

Algorithm 1 LA-UPAHEM.

Input: User preferences $\mathbf{P}$, task-specific context $\mathbf{C} = (\mathbf{c}_{\text{des}}, \mathbf{c}_{\text{kpi}}, \mathbf{c}_{\text{rref}}, \mathbf{c}_{\text{mref}})$, prompt templates $\mathbf{T} = (\mathbf{t}_{\text{code}}, \mathbf{t}_{\text{analysis}}, \mathbf{t}_{\text{search}})$, pre-trained policy parameters θ_0.
Parameters: Outer iteration count ι, candidate count per iteration n.
Output: Optimal preference-aligned policy $\pi^{\star}_{\mathbf{P}}$ with corresponding $(\mathcal{R}^{\star}_{\mathbf{P}}, \mathcal{M}^{\star}_{\mathbf{P}})$.

Initialize memory pool $\mathcal{D} \leftarrow \emptyset$, policy parameters $\theta \leftarrow \theta_0$
for $i = 1$ **to** ι **do**
  $\{(\mathcal{R}^{j}_{\mathbf{P}}, \mathcal{M}^{j}_{\mathbf{P}})\}_{j=1}^{n} \leftarrow \mathcal{F}_{\text{code}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{code}}, \mathcal{D}_{i-m:i-1})$ {$\mathcal{D}_{i-m:i-1} = \emptyset$ when $i = 1$}
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathcal{R}^{j}_{\mathbf{P}}, \mathcal{M}^{j}_{\mathbf{P}})\}_{j=1}^{n}$
  for $j = 1$ **to** n **do**
    Create environments $\text{env}^{\text{train}}_{j}$, $\text{env}^{\text{test}}_{j}$ using $(\mathcal{R}^{j}_{\mathbf{P}}, \mathcal{M}^{j}_{\mathbf{P}})$
    $\pi^{\star}_{j} \leftarrow \text{SAC}(\pi_{\theta}, \text{env}^{\text{train}}_{j})$
    $(\mathbf{k}_j, \mathbf{u}_j) \leftarrow \text{Evaluate}(\pi^{\star}_{j}, \text{env}^{\text{test}}_{j})$
  end for
  $\mathbf{e}_i \leftarrow \{(\mathbf{k}_j, \mathbf{u}_j)\}_{j=1}^{n}$; $\mathcal{D} \leftarrow \mathcal{D} \cup \mathbf{e}_i$
  $\mathbf{f}_i \leftarrow \mathcal{F}_{\text{analysis}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{analysis}}, \mathbf{e}_i, \{(\mathcal{R}^{j}_{\mathbf{P}}, \mathcal{M}^{j}_{\mathbf{P}})\}_{j=1}^{n})$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathbf{f}_i$
  $j^{\star} \leftarrow \mathcal{F}_{\text{search}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{search}}, \mathbf{e}_i)$
  $\theta \leftarrow \theta(\pi^{\star}_{j^{\star}})$
end for
$(i^{\star}, j^{\star}) \leftarrow \mathcal{F}_{\text{search}}(\mathbf{P}, \mathbf{C}, \mathbf{t}_{\text{search}}, \{\mathbf{e}_i\}_{i=1}^{\iota})$
return $\pi^{\star}_{\mathbf{P}} \leftarrow \pi^{(i^{\star})}_{j^{\star}}$, $\mathcal{R}^{\star}_{\mathbf{P}} \leftarrow \mathcal{R}^{i^{\star},j^{\star}}_{\mathbf{P}}$, $\mathcal{M}^{\star}_{\mathbf{P}} \leftarrow \mathcal{M}^{i^{\star},j^{\star}}_{\mathbf{P}}$

helps the policy adapt to new reward landscapes without collapsing into suboptimal local minima. Second, as an off-policy algorithm, SAC can reuse experiences from a replay buffer, enabling efficient warm-start from previous iterations and reducing sample complexity during iterative refinement.

5.4. Adaptive iterative loop

LA-UPAHEM integrates the LLM agents with DRL training through a nested iterative loop, progressively refining the preference-aware policy with minimal human intervention. The outer loop (iterations $i = 1, \ldots, \iota$) manages LLM-agent interactions, code generation, result analysis, and feedback exchange, while each outer iteration generates n candidate $(\mathcal{R}_{\mathbf{P}}, \mathcal{M}_{\mathbf{P}})$ pairs. The inner loop (candidates $j = 1, \ldots, n$) instantiates an MDP environment for each candidate, trains a policy using SAC, and evaluates performance in the test environment.

The memory pool $\mathcal{D}$ accumulates structured records across all iterations: $\mathcal{D} = \bigcup_{i=1}^{\iota}\{\{(\mathcal{R}^{i,j}_{\mathbf{P}}, \mathcal{M}^{i,j}_{\mathbf{P}})\}_{j=1}^{n}, \mathbf{e}_i, \mathbf{f}_i\}$. As described in Eq. (13), the Code Generation Agent retrieves the most recent m iterations (default $m = 3$) via a deterministic sliding window $\mathcal{D}_{i-m:i-1}$. To reduce token consumption, trajectory data within $\mathbf{e}_i$ is excluded from the Code Generation Agent's retrieval and is provided only to the Result Analysis Agent. Algorithm 1 formalizes the complete procedure.

LA-UPAHEM employs two complementary evaluation mechanisms. The inner loop evaluates policies on fixed KPIs, providing objective performance metrics. The outer loop leverages LLM agents to assess preference alignment holistically via $\mathcal{F}_{\text{analysis}}$ and $\mathcal{F}_{\text{search}}$, accounting for qualitative aspects that raw KPIs may not capture, such as whether conditional rules are satisfied. At each outer iteration, the best policy from the previous iteration initializes training for all n candidates (warm-start), accelerating convergence and improving stability compared to training from scratch.

In practical deployment, the outer loop involving LLM agents is invoked only when user preferences change, which occurs infrequently in typical residential settings. During normal operation, only the trained DRL policy from the inner loop is executed for real-time HEMS control, incurring minimal computational overhead suitable for household environments. When preferences do change, the LLM agents can be accessed through cloud APIs without requiring local computational resources. Furthermore, since the framework outputs explicit reward and state modification functions $(\mathcal{R}^{\star}_{\mathbf{P}}, \mathcal{M}^{\star}_{\mathbf{P}})$, these can be reused for policy transfer or retraining without invoking LLM agents again, enabling efficient adaptation to new environments or device configurations.

Compared with conventional DRL-based HEMS that rely on fixed, manually designed reward functions, LA-UPAHEM provides two distinguishing properties: (i) the nested iterative loop gradually refines LLM-generated optimization signals, balancing exploration across diverse candidates with exploitation through warm-start fine-tuning; and (ii) all generated components—reward functions $(\mathcal{R}_{\mathbf{P}})$, state modifiers $(\mathcal{M}_{\mathbf{P}})$, and iterative feedback $\mathbf{f}_i$—are human-readable, facilitating inspection, debugging, and user trust.

5.5. Privacy considerations

Since LA-UPAHEM transmits user context to cloud-based LLM APIs during the iterative loop, privacy implications deserve explicit consideration. Three categories of data leave the household: (i) user preference statements $\mathbf{P}$, which may reveal lifestyle habits (e.g., "I work night shifts"); (ii) household context $\mathbf{C}$, containing device specifications and occupancy patterns; and (iii) memory-pool records $\mathcal{D}_{i-m:i-1}$, which include KPI trajectories that can expose energy consumption profiles. Importantly, data transmission is confined to the outer-loop iterations and is entirely absent during real-time policy execution, where the trained DRL policy runs locally. For a typical household that updates preferences a few times per year, the exposure window is limited to a small number of API calls (e.g., 15 calls per preference update with $\iota = 5$ iterations).

Several practical mitigations can be adopted without modifying the core framework. First, data minimization can strip personally identifiable information from $\mathbf{C}$ (e.g., replacing absolute timestamps with relative offsets); the current implementation already excludes raw trajectories from Code Generation Agent prompts to reduce information leakage (Section 5.4). Second, retention limits can purge memory-pool records beyond the sliding window m, and zero-retention policies can be requested from LLM API providers. Third, deploying open-source LLMs (e.g., LLaMA-3 70B) on-premises eliminates cloud transmission entirely, with 70B-class models already achieving acceptable preference compliance in English (Section 6.7). Fourth, query anonymization can replace real device names and locations with generic identifiers before API calls.

6. Experiments

We empirically evaluate LA-UPAHEM on real-world residential energy data. The experiments assess four aspects: (1) preference alignment quality relative to baselines across all eight preference scenarios; (2) the necessity of iterative refinement and the contribution of each LLM agent module; (3) robustness to environmental noise, cross-environment variation, evaluation weight perturbations, and linguistic paraphrasing; and (4) generality across DRL algorithms, LLM backbones, and preference languages.

6.1. Experimental settings

We instantiate the HEMS simulation using real-world traces from Pecan Street [38], NOAA [39], and CityLearn [40]. Detailed data sources, fields, and preprocessing steps are summarized in Appendix B (Table 13); household environment parameters are provided in Appendix A (Table 12).

As illustrated in Fig. 2, the framework naturally separates training, evaluation, and deployment environments. In a physical deployment, the iterative optimization phase would use a real building as the training environment, making data leakage inapplicable. Since we validate the method in simulation, we ensure equivalent isolation by randomly

Table 3
Baseline comparison. "Macro" and "Micro" indicate access to macro-level weights $\boldsymbol{\alpha}^\star$ or micro-level rules $\mathcal{G}^\star$ (defined in Section 6.3); **P** means derived from natural-language preferences. "Iterative" denotes whether the method refines outputs across rounds. "State aug." indicates whether the method augments the state space $\mathcal{S}$.

Method	Macro	Micro	Iterative	State aug.	LLM Usage
D1: Fixed-Reward DRL	✗	✗	✗	✗	None
D2: Oracle-Macro DRL	$\boldsymbol{\alpha}^\star$	✗	✗	✗	None
D3: Oracle-Full DRL	$\boldsymbol{\alpha}^\star$	$\mathcal{G}^\star$	✗	✗	None
H1: Static RL + LLM	**P**	**P**	✗	**P**	Once before training
H2: LLM-Reward [31]	**P**	✗	✗	✗	Per episode
H3: LLM-Action Filter [29]	✗	**P**	online	✗	Per timestep
LA-UPAHEM (ours)	**P**	**P**	✓	**P**	Per iteration

partitioning the available days into three disjoint subsets, each instantiating an independent simulation environment: (i) training days drive the inner-loop DRL policy optimization; (ii) validation days are used exclusively by the outer-loop LLM agents—the KPI values and control trajectories stored in the memory pool D and analyzed by the Result Analysis Agent all come from validation episodes; (iii) test days simulate real-world deployment and are never accessed by either the LLM agents or the DRL training process. All results are averaged over 10 independent random seeds per preference.

The control MDP operates with $\Delta t = 5$ min timesteps and one-day episodes ($\tau = 288$ steps). We define eight prototypical preferences $\mathbf{P}_{1:8}$ covering macro-level trade-offs ($\mathbf{P}_{1:3}$), micro-level rules ($\mathbf{P}_{4:6}$), compositional constraints ($\mathbf{P}_7$), and a preference-agnostic baseline ($\mathbf{P}_8$), as detailed in Table 2.

LA-UPAHEM generates $n = 5$ candidate $(\mathcal{R}_\mathbf{P}, \mathcal{M}_\mathbf{P})$ pairs per outer iteration for $\iota = 5$ iterations, with each candidate trained for 90 episodes. Unless otherwise specified, we use SAC [41] as the DRL backend with: actor-critic MLPs of hidden dimensions $[256, 512, 64, 512, 256]$, ReLU activations, Adam optimizer ($\mathrm{lr} = 3 \times 10^{-4}$), discount factor $\gamma = 0.99$, Polyak coefficient $\tau_{\text{polyak}} = 0.005$, automatic entropy tuning, batch size 256, and replay buffer capacity 10^6. The LLM backbone is DeepSeek V3 (671B) [42] with temperature 0.7 for code generation and 0 for evaluation parsing. Experiments run on Ubuntu 20.04 with Python 3.10, PyTorch 2.0, an Intel i7-14600K CPU, and an NVIDIA RTX 4090 GPU.

6.2. Baselines

We compare LA-UPAHEM against classical DRL methods and representative RL + LLM integration paradigms (Table 3).

Classical baselines. D1 (**Fixed-Reward DRL**) trains a standard DRL agent with fixed, preference-agnostic reward weights. D2 (**Oracle-Macro DRL**) and D3 (**Oracle-Full DRL**) use the linear reward formulation $r_t = \sum_{k\in\{\mathrm{c,t,e}\}} \alpha_k \mathcal{R}_k$ from Eq. (12), where D2 substitutes oracle-specified macro weights $\boldsymbol{\alpha}^\star$ matching the evaluation target, and D3 further incorporates handcrafted micro-level rules $\mathcal{G}^\star$ as additional reward penalties (Table 4). For purely macro-level preferences ($\mathbf{P}_{1:3}$), D2 and D3 are equivalent since no micro-level rules apply. These baselines require no LLM but demand manual specification for each preference.

RL + LLM baselines. H1 (**Static RL + LLM**) parses **P** into structured parameters via a single LLM call before training, without subsequent verification. H2 (**LLM-Reward** [31]) generates executable reward code from **P** and queries the LLM after each training episode to regenerate reward variants, but lacks a dedicated analysis agent to leverage performance feedback. H3 (**LLM-Action Filter** [29]) bypasses reward design entirely: it uses a preference-agnostic DRL policy (equivalent to D1) and queries the LLM at every control timestep to filter actions violating language-expressed rules. This provides online feedback but couples LLM availability to real-time operation.

In contrast, LA-UPAHEM queries the LLM only at the start of each outer iteration to generate and refine $(\mathcal{R}_\mathbf{P}, \mathcal{M}_\mathbf{P})$ pairs. It combines: (i) iterative refinement based on policy evaluation feedback $\mathbf{e}_i$, (ii) joint generation of reward functions and state modification functions, and (iii) complete decoupling of LLM inference from real-time control—a combination not found in existing RL + LLM methods for energy management.

Table 4
Canonical evaluation targets for each preference: macro weights $\boldsymbol{\alpha}^\star = (\alpha_\mathrm{c}, \alpha_\mathrm{t}, \alpha_\mathrm{e})$ (sum to 1) and micro-level rules $\mathcal{G}^\star$. "—" indicates no micro-level rule applies.

	α_c	α_t	α_e	Rule $g_j(s_t, a_t)$
$\mathbf{P}_1$ (Cost-prio.)	0.62	0.18	0.20	–
$\mathbf{P}_2$ (Emis.-prio.)	0.10	0.15	0.75	–
$\mathbf{P}_3$ (Comfort-prio.)	0.15	0.70	0.15	–
$\mathbf{P}_4$ (No home-chg.)	0.34	0.33	0.33	$\mathrm{occ}_t{=}1 \implies P_t^\mathrm{b} \le 0$
$\mathbf{P}_5$ (AC home & night)	0.34	0.33	0.33	$\neg(\mathrm{occ}_t{=}1 \wedge \mathrm{hr}_t{\in}[22,7]) \implies M_t^\mathrm{a}{=}0$
$\mathbf{P}_6$ (Peak disch.)	0.34	0.33	0.33	$\kappa_t < \kappa^\mathrm{peak} \implies P_t^\mathrm{b} \ge 0$
$\mathbf{P}_7$ (Tolerate cold)	0.52	0.23	0.25	$T_t^\mathrm{in} < T_t^\mathrm{set} \implies M_t^\mathrm{a}{=}0$
$\mathbf{P}_8$ (No pref.)	0.34	0.33	0.33	–

6.3. Evaluation specification

To fairly assess preference alignment, we define evaluation targets that are independent of the training process. Each method is scored against canonical macro weights $\boldsymbol{\alpha}^\star$ and micro-level rules $\mathcal{G}^\star$ derived from natural-language preferences **P**.

Canonical targets. Following the "LLM-as-a-Judge" paradigm widely adopted in natural language processing evaluation [43], we use a deterministic evaluation parser Φ_eval to simulate consistent user interpretation of preferences. Specifically, Φ_eval is an LLM-based parser with the temperature fixed at 0, representing an idealized user who interprets natural-language preferences **P** into quantifiable targets:

$$(\boldsymbol{\alpha}^\star, \mathcal{G}^\star) = \Phi_\text{eval}(\mathbf{P}, \mathbf{C}, \mathbf{t}_\text{eval}), \tag{17}$$

where $\mathbf{t}_\text{eval}$ is a prompt template that guides the LLM to interpret **P** within the HEMS context and output normalized weights ($\sum_k \alpha_k^\star = 1$) and logical predicates. The resulting $\boldsymbol{\alpha}^\star = (\alpha_\mathrm{c}^\star, \alpha_\mathrm{t}^\star, \alpha_\mathrm{e}^\star)$ are nonnegative weights satisfying $\sum_k \alpha_k^\star = 1$, and $\mathcal{G}^\star = \{g_j\}$ is a set of logical predicates encoding constraints such as occupancy gating or time-of-use restrictions (Table 4). The deterministic temperature ensures that canonical targets are fully reproducible given the same prompt and LLM version. Importantly, Φ_eval is used only for test-time scoring and never constrains reward construction during training. We emphasize that Φ_eval serves as a "simulated user" to provide ground-truth interpretation of preferences, which is fundamentally different from LA-UPAHEM's code generation agents: (i) Φ_eval uses a distinct prompt template $\mathbf{t}_\text{eval}$ that outputs structured weights and rules, whereas LA-UPAHEM agents use $\mathbf{T} = (\mathbf{t}_\text{code}, \mathbf{t}_\text{analysis}, \mathbf{t}_\text{search})$ to generate executable code; (ii) Φ_eval-derived targets are applied equally to evaluate all methods, including non-LLM baselines D1–D3, ensuring fair comparison; (iii) Oracle baselines D2/D3 directly receive these targets as privileged input, giving them an advantage over LA-UPAHEM, which must infer appropriate reward structures from scratch.

Weighted Improvement Ratio (WIR). To evaluate macro-level preference alignment independently of the comparison set, we define WIR relative to a fixed reference baseline $\mathcal{L}_k^\mathrm{ref}$. Specifically, $\mathcal{L}_k^\mathrm{ref}$ is the mean KPI of D1 (Fixed-Reward DRL, which is entirely preference-agnostic) across all eight preferences $\mathbf{P}_{1:8}$:

$$\mathrm{WIR}^{(m)}(\mathbf{P}) = \sum_{k\in\{\mathrm{c,t,e}\}} \alpha_k^\star \frac{\mathcal{L}_k^\mathrm{ref} - \mathcal{L}_k^{(m)}}{\mathcal{L}_k^\mathrm{ref}}, \qquad \mathcal{L}_k^\mathrm{ref} = \frac{1}{8}\sum_{p=1}^{8} \mathcal{L}_k^{(\mathrm{D1},\mathbf{P}_p)}. \tag{18}$$

Table 5
Raw KPI values (mean ± std over random seeds; lower is better). $\mathcal{L}_c$: electricity cost ($); $\mathcal{L}_t$: thermal discomfort (h); $\mathcal{L}_e$: carbon emissions (kg CO_2, scaled ×10). Bold = best per KPI–preference pair. Under $\mathbf{P}_{1-3}$ and $\mathbf{P}_8$, D2 ≡ D3 (no micro-level rules); under $\mathbf{P}_8$ both further reduce to D1 (identical balanced weights). Derived WIR and RCR metrics are visualized in Figs. 3 and 4.

Method	$\mathbf{P}_1$ (Cost-prio.)			$\mathbf{P}_2$ (Emission-prio.)			$\mathbf{P}_3$ (Comfort-prio.)			$\mathbf{P}_4$ (No home-charging)		
	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$
D2: Oracle-Macro DRL	7.21±0.10	4.45±0.08	1.18±0.04	8.39±0.12	3.75±0.07	1.12±0.03	9.38±0.13	1.82±0.03	**1.74**±0.05	**9.15**±0.13	2.78±0.05	**1.62**±0.05
D3: Oracle-Full DRL	≡ D2 (no micro-level rules for $\mathbf{P}_{1-3}$)									10.95±0.16	2.59±0.05	2.06±0.07
H1: Static RL + LLM	8.25±0.26	**3.98**±0.16	1.44±0.09	8.42±0.27	**3.71**±0.15	1.22±0.07	**8.91**±0.29	2.24±0.09	1.87±0.11	10.31±0.33	2.88±0.12	2.13±0.13
H2: LLM-Reward	7.15±0.20	4.89±0.19	1.18±0.06	7.95±0.22	3.95±0.15	1.09±0.06	9.41±0.26	2.03±0.08	1.78±0.10	10.25±0.29	2.81±0.11	2.05±0.11
H3: LLM-Action Filter	7.43±0.18	4.55±0.15	1.22±0.06	8.52±0.20	3.83±0.12	1.28±0.06	9.45±0.23	2.12±0.07	1.86±0.09	11.17±0.27	**2.45**±0.08	2.15±0.10
LA-UPAHEM (Ours)	**6.65**±0.12	4.35±0.10	**1.01**±0.04	**7.73**±0.14	4.32±0.10	**0.98**±0.04	9.71±0.17	**1.57**±0.04	1.82±0.07	10.79±0.19	2.55±0.06	2.08±0.08
Method	$\mathbf{P}_5$ (AC home & night)			$\mathbf{P}_6$ (Peak discharge)			$\mathbf{P}_7$ (Tolerate cold)			$\mathbf{P}_8$ (No preference)		
	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$
D2: Oracle-Macro DRL	8.67±0.12	**2.58**±0.05	1.52±0.05	8.35±0.12	2.52±0.05	**1.45**±0.04	7.17±0.10	**5.05**±0.09	1.32±0.04	≡ D1		
D3: Oracle-Full DRL	6.21±0.09	7.48±0.15	1.18±0.04	8.72±0.13	**2.42**±0.05	1.58±0.05	6.05±0.09	8.29±0.17	**0.91**±0.03	≡ D1		
H1: Static RL + LLM	7.35±0.24	6.47±0.26	1.26±0.08	8.65±0.28	2.62±0.10	1.65±0.10	6.35±0.20	8.21±0.33	1.08±0.06	8.68±0.28	2.88±0.12	1.75±0.10
H2: LLM-Reward	6.65±0.19	8.65±0.33	1.14±0.06	8.42±0.24	2.55±0.10	1.55±0.09	6.25±0.18	8.15±0.31	1.15±0.06	8.45±0.24	2.70±0.10	1.64±0.09
H3: LLM-Action Filter	**5.82**±0.14	9.38±0.30	**1.09**±0.05	8.78±0.21	2.48±0.08	1.70±0.08	6.15±0.15	8.05±0.26	1.12±0.05	8.55±0.20	2.78±0.08	1.70±0.08
LA-UPAHEM (Ours)	6.12±0.11	8.45±0.20	**1.09**±0.04	**8.15**±0.15	2.45±0.06	1.50±0.06	**5.75**±0.10	8.11±0.19	1.01±0.04	8.25±0.15	2.55±0.06	1.56±0.06
Preference-Agnostic Baseline												
D1: Fixed-Reward DRL	(Avg. over $\mathbf{P}_{1:8}$)			$\mathcal{L}_c$: 8.10 ± 0.16	$\mathcal{L}_t$: 2.49 ± 0.21	$10\mathcal{L}_e$: 1.49 ± 0.15						

Because each KPI is normalized independently by its reference value, WIR captures relative (percentage) improvements: a 1% change in any KPI k contributes exactly $\alpha_k^\star \times 1\%$ to WIR. Positive WIR indicates a net weighted improvement over the preference-agnostic baseline, while negative WIR reveals that relative deterioration in some KPIs outweighs relative gains in others. This can occur even when $\boldsymbol{\alpha}^\star$ are set correctly, because different KPIs exhibit different relative elasticities under policy optimization (e.g., thermal discomfort can easily double under cost-prioritized policies, whereas cost reductions remain modest). We analyze the implications of this asymmetry in Section 6.4. We report WIR averaged over macro-level preferences $\mathbf{P}_{1:3}$ as the primary macro-alignment metric, because $\mathbf{P}_{4:7}$ impose micro-level behavioral rules whose enforcement inherently worsens macro-level KPI balance—an expected and desirable trade-off captured by the Rule Compliance Rate below.

Rule Compliance Rate (RCR). The micro-level metric measures compliance with behavioral rules, conditioned on the timesteps where each rule is applicable:

$$\mathrm{RCR}^{(m)} = \begin{cases} 1 - \dfrac{1}{|\mathcal{G}^\star|} \displaystyle\sum_{j=1}^{|\mathcal{G}^\star|} \frac{V_j^{(m)}}{\tau_j^{(m)}}, & |\mathcal{G}^\star| > 0, \\ 1, & |\mathcal{G}^\star| = 0, \end{cases} \qquad V_j^{(m)} = \sum_{t=1}^{\tau} \mathbb{1}\left\{ g_j(s_t^{(m)}, a_t^{(m)}) = 0 \right\}, \tag{19}$$

where $\tau_j^{(m)}$ is the number of timesteps at which rule g_j's precondition is satisfied (e.g., the occupant is home for $\mathbf{P}_4$), and $V_j^{(m)}$ is the number of violations among those timesteps. Preferences without explicit micro-level rules ($|\mathcal{G}^\star| = 0$, i.e., $\mathbf{P}_{1:3}$ and $\mathbf{P}_8$) yield RCR = 1 by definition. Higher values indicate better compliance.

Additional metrics. We also report: (i) raw KPI values (cost $\mathcal{L}_c$, discomfort $\mathcal{L}_t$, emissions $\mathcal{L}_e$; mean±std across seeds); (ii) failure rate, the fraction of runs producing invalid policies (e.g., violating device constraints, degenerate behavior, or training divergence); and (iii) convergence speed measured by training steps to reach stable performance.

6.4. Preference alignment performance

Table 5 reports raw KPI values across all preference categories, while Figs. 3 and 4 visualize the derived WIR and RCR metrics, and Figs. 5 and 6 provide complementary perspectives on preference alignment.

For macro-level preferences ($\mathbf{P}_1$–$\mathbf{P}_3$) that specify KPI trade-offs, LA-UPAHEM demonstrates clear adaptive behavior. Under cost-prioritized $\mathbf{P}_1$, it achieves the lowest $\mathcal{L}_c = 6.65$; under comfort-prioritized $\mathbf{P}_3$, it shifts to $\mathcal{L}_t = 1.57$—the lowest discomfort among all methods. This contrasts with D1, which produces nearly identical outputs regardless of preference ($\mathcal{L}_c \approx 8.1$, $\mathcal{L}_t \approx 2.5$ across all $\mathbf{P}$). Fig. 3 shows WIR across methods: LA-UPAHEM consistently achieves the highest values with lower variance than H1/H2, as our iterative refinement progressively corrects initial LLM misinterpretations through feedback $\mathbf{f}_i$. The reduced variance is particularly important for practical deployment, as it indicates that LA-UPAHEM produces reliable preference alignment across different random seeds and environmental configurations.

Notably, even Oracle baseline D2—which receives the ground-truth preference weights $\boldsymbol{\alpha}^\star$—exhibits negative WIR under $\mathbf{P}_1$. This is not a metric artifact but reveals a fundamental challenge: reward-function weights do not translate linearly into proportional KPI improvements. Although the exponential reward transforms (Section 5.1) rescale individual reward components to a common range, the mapping from reward weights to actual policy outcomes remains highly nonlinear, governed by the environment dynamics (e.g., thermal inertia makes comfort far more elastic than cost in relative terms), the structure of the action space, and the DRL optimizer's convergence behavior. Consequently, applying the same $\boldsymbol{\alpha}^\star$ to both the reward function and WIR evaluation does not guarantee positive WIR. This finding underscores the core motivation of LA-UPAHEM: rather than relying on a one-shot weight specification, the iterative feedback loop (Section 5.4) enables the LLM agents to observe actual control outcomes and progressively adjust the reward structure until the realized KPI balance aligns with user intent.

Micro-level and compositional preferences ($\mathbf{P}_4$–$\mathbf{P}_7$) impose behavioral constraints beyond KPI weights. Fig. 4 shows LA-UPAHEM achieving the highest RCR with fewest rule violations. Notably, enforcing micro-level rules inherently constrains the action space—for example, $\mathbf{P}_4$'s no-charging-at-home rule limits BESS scheduling flexibility, resulting in higher $\mathcal{L}_c$ for all compliant methods. This trade-off is expected: the purpose of micro-level preferences is to enforce user-specified operational constraints, even at the expense of KPI optimality. Fig. 5 illustrates how LA-UPAHEM dynamically adapts device operation to satisfy these constraints: under $\mathbf{P}_4$ (no battery charging at home),

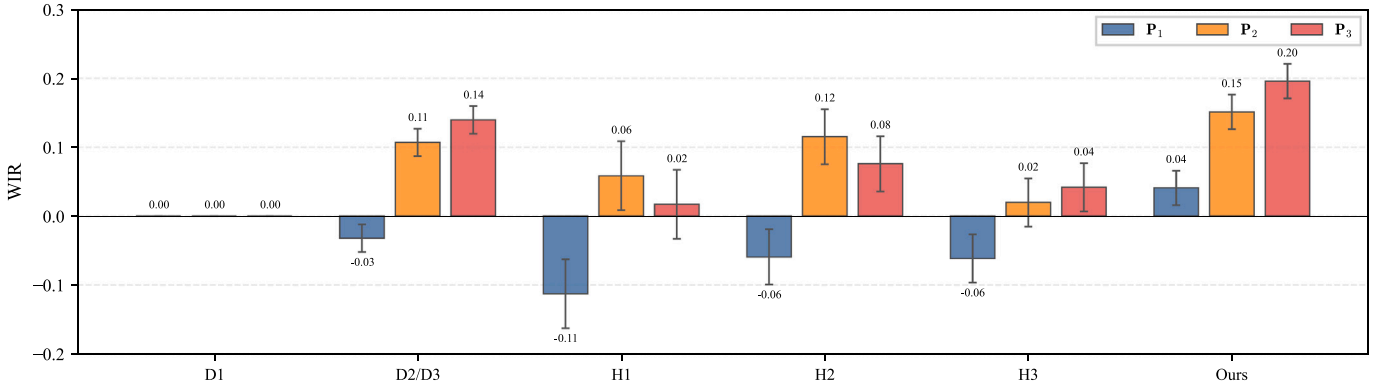


Fig. 3. Weighted Improvement Ratio (WIR) under macro-level preferences $\mathbf{P}_{1:3}$, derived from Table 5. Higher WIR indicates better preference alignment relative to D1 (whose WIR = 0 by definition). Error bars show the standard deviation across random seeds.

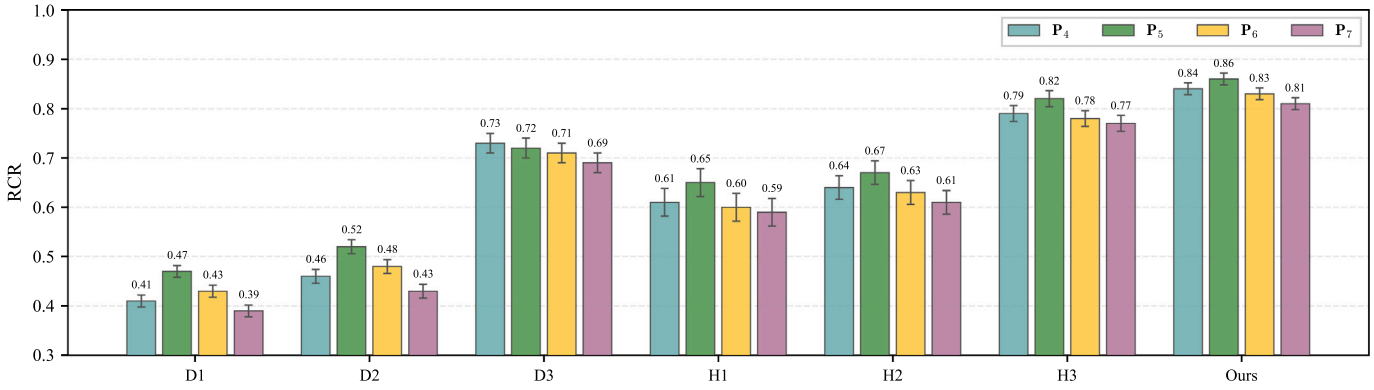


Fig. 4. Rule compliance rate (RCR) under behavioral rule preferences $\mathbf{P}_{4:7}$. Higher RCR indicates fewer rule violations. Error bars indicate standard deviation across random seeds.

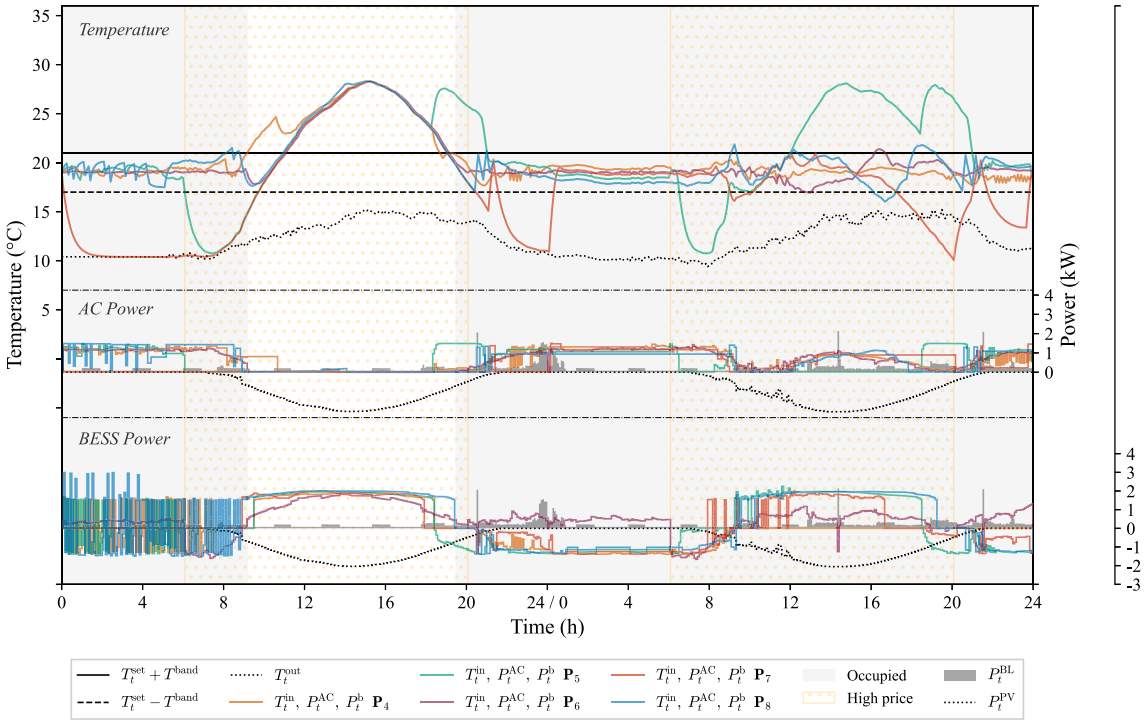


Fig. 5. Control trajectories of AC and BESS under micro-level preferences $\mathbf{P}_{4\text{–}6}$, compositional $\mathbf{P}_7$, and baseline $\mathbf{P}_8$.

LA-UPAHEM learns to suppress charging when $\text{occ}_t = 1$, whereas D2 continues charging regardless. H3 achieves competitive RCR for $\mathbf{P}_5$, but at a cost: its reactive filtering yields $\mathcal{L}_t = 9.38$—higher than LA-UPAHEM's 8.45—because blocking actions without considering downstream consequences causes cumulative discomfort. LA-UPAHEM avoids this by internalizing rules into the reward function during training, enabling the policy to anticipate and plan around constraints rather than reactively comply.

To concretely illustrate how this internalization works, we present a representative $(\mathcal{M}_{\mathbf{P}_5}, \mathcal{R}_{\mathbf{P}_5})$ pair generated by the Code Generation Agent for $\mathbf{P}_5$ ("*Only use AC at night when I'm home*"):

```
# -- StateModifier (excerpt) --
modified_state = original_state.copy()
hour = original_state['hour_history'][-1]
occ  = original_state['occupancy_history'][-1]
# Novel feature: night-and-home indicator (absent from all templates)
is_night_home = float(occ == 1 and (hour >= 22 or hour <= 7))
modified_state['is_night_home_history'] = [is_night_home]
# Novel feature: conditional comfort urgency
temp_in  = original_state['indoor_temperature_history'][-1]
temp_set = original_state['target_temperature_history'][-1]
comfort_urgency = max(0, abs(temp_in - temp_set) - 1.0) * is_night_home
modified_state['comfort_urgency_history'] = [comfort_urgency]

# -- Reward (excerpt, weights: night_ac=3.0, comfort=1.5, cost=0.5) --
is_night_home   = state['is_night_home_history'][-1]
comfort_urgency = state['comfort_urgency_history'][-1]
ac_mode = state['ac_mode_history'][-1]
cost        = state['home_electrical_cost_history'][-1]
# Penalize AC usage outside the night-home window
ac_penalty    = -3.0 * abs(ac_mode) * (1.0 - is_night_home)
comfort_reward = -1.5 * comfort_urgency
cost_reward   = -0.5 * cost
reward = ac_penalty + comfort_reward + cost_reward
```

Three observations stand out. First, the LLM invented two state features absent from the reference templates $(\mathcal{M}_c, \mathcal{M}_t, \mathcal{M}_e)$: `is_night_home`—a compositional binary indicator fusing occupancy with time-of-day—and `comfort_urgency`, which conditions thermal discomfort on this indicator, demonstrating genuine feature invention beyond template reparameterization. Second, the reward uses linear weighted penalties instead of the templates' exponential transforms ($\mathcal{R}_c = \exp(\mathcal{M}_c)$, etc.), reflecting the flexibility of LLM-based code generation. We note that across iterations, exponential transforms gradually become dominant as the Result Analysis Agent identifies smoother gradient landscapes as beneficial for training stability. Third, the generated code provides an interpretable bridge between natural language preferences and DRL objectives: practitioners can directly inspect which features encode the preference semantics and how the reward trades off competing objectives.

Compared with Oracle baselines, LA-UPAHEM is competitive on raw KPIs despite lacking privileged access to ground-truth specifications. Under $\mathbf{P}_7$, it achieves $\mathcal{L}_c = 5.75$ versus D3's 6.05, benefiting from state augmentation: $\mathcal{M}_{\mathbf{P}}$ explicitly encodes preference-relevant features (e.g., occupancy-conditioned indicators) into the augmented state $\tilde{s}_t$, reducing the agent's credit assignment burden compared with Oracle baselines that operate on the raw state space. In other scenarios Oracle baselines achieve marginally better individual KPIs (e.g., D3's $\mathcal{L}_e = 0.091$ under $\mathbf{P}_7$ vs. 0.101), but they require domain experts to manually specify weights and rules for each preference—a process that does not scale to diverse real-world needs.

The radar plots in Fig. 6 summarize normalized KPI scores across all eight preferences. LA-UPAHEM exhibits the most balanced profiles, demonstrating consistent adaptation without preference-specific training data or manually crafted reward functions.

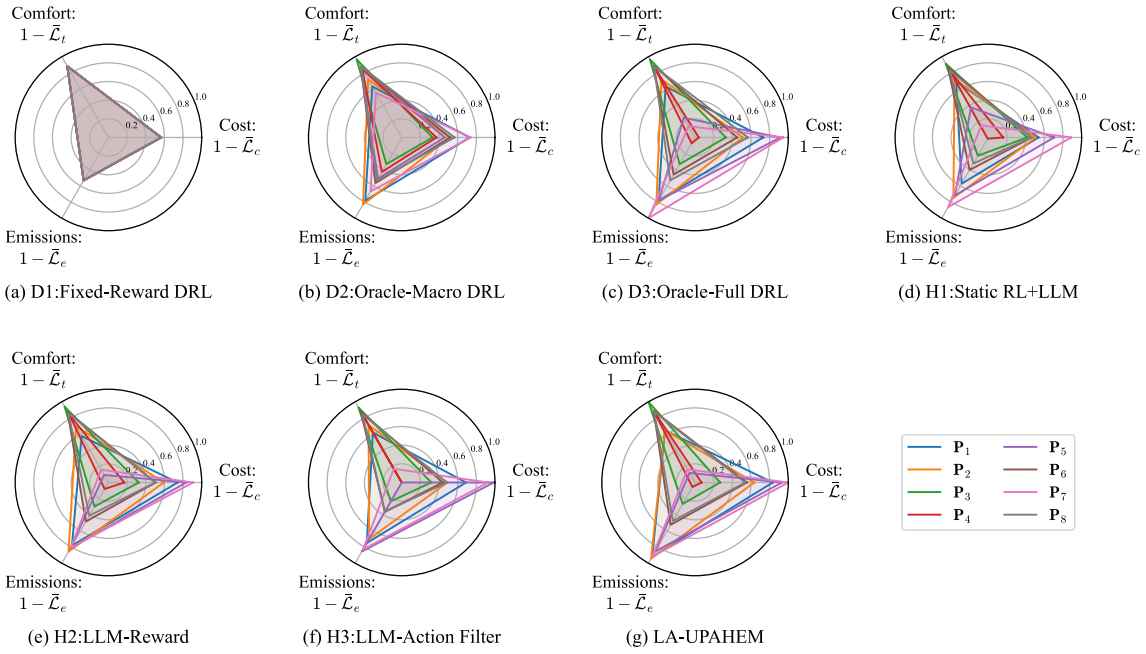
For the compositional preference $\mathbf{P}_7$, which combines a cost-saving macro trade-off with conditional AC suppression, Fig. 5 provides visual evidence: LA-UPAHEM correctly suppresses AC whenever $T_t^{\text{in}} < T_t^{\text{set}}$, directly implementing the rule at the policy level. In contrast, H3 enforces the same rule reactively via per-timestep action masking, which causes $\mathcal{L}_t = 8.05$—substantially higher discomfort—because blocking individual actions without foresight prevents the policy from planning around the constraint.

6.5. Iterative mechanism and ablation

A fundamental question for LLM-augmented RL is whether iterative refinement justifies its complexity. We compare LA-UPAHEM against H1 (Static RL + LLM, which interprets preferences once without feedback) and two ablation variants: "w/o Feedback" removes the Result Analysis Agent (no corrective feedback $\mathbf{f}_i$), and "w/o Warm-start" removes the Optimal Performance Search Agent (each iteration trains from scratch).

Iteration is essential because natural language preferences are inherently ambiguous and, as discussed in Section 6.4, the mapping from reward weights to policy outcomes is highly nonlinear. H1 illustrates this: its one-shot interpretation of $\mathbf{P}_3$ yields $\mathcal{L}_t = 2.24$—far worse than LA-UPAHEM's 1.57—and its failure rate (34.3%, Fig. 8) is 6× higher than LA-UPAHEM's (5.4%), because static approaches cannot recover from initial misinterpretations.

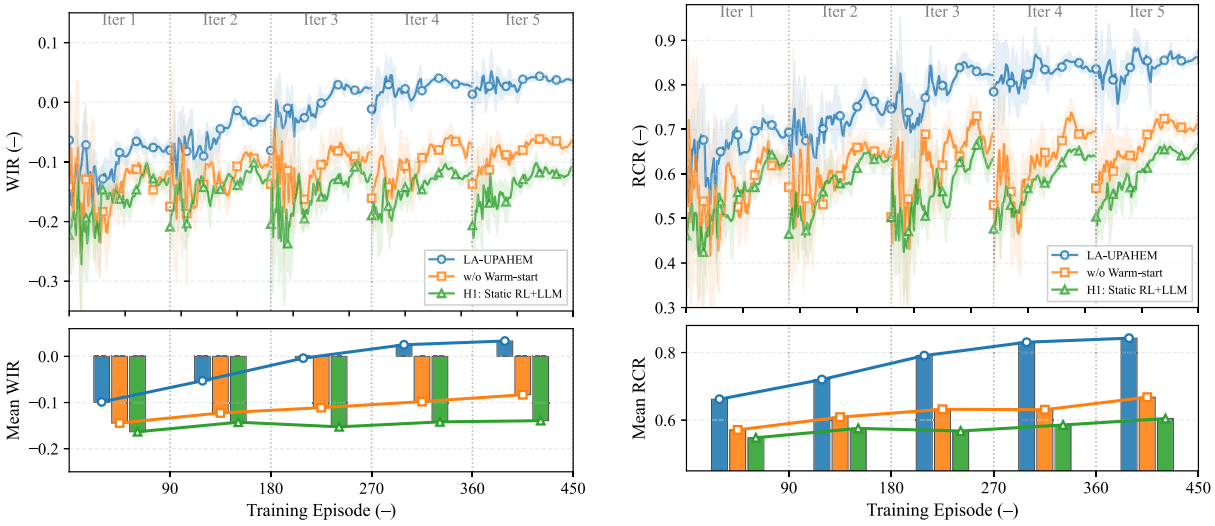
Fig. 7 validates the closed-loop design. In both panels, LA-UPAHEM exhibits consistent upward trajectories: under $\mathbf{P}_1$, WIR increases steadily across iterations as the reward function is progressively refined; the micro-level convergence (Fig. 7)(b) shows a similar pattern for $\mathbf{P}_5$ as early reward functions progressively strengthen penalties based on observed violation rates. In contrast, H1 produces nearly flat curves



(a) D1:Fixed-Reward DRL (b) D2:Oracle-Macro DRL (c) D3:Oracle-Full DRL (d) H1:Static RL+LLM

(e) H2:LLM-Reward (f) H3:LLM-Action Filter (g) LA-UPAHEM

Fig. 6. Radar plots of normalized KPI scores under each **P**. Each axis shows $1-\bar{\mathcal{L}}_k$ where $\bar{\mathcal{L}}_k = (\mathcal{L}_k - \mathcal{L}_k^{\min})/(\mathcal{L}_k^{\max} - \mathcal{L}_k^{\min})$ is the min-max normalization of KPI k across all method–preference pairs; higher values indicate better performance.



(a) WIR under $\mathbf{P}_1$ (b) RCR under $\mathbf{P}_5$

Fig. 7. Convergence of preference alignment during training. In the upper panels, each curve shows the training evaluation trajectory of the iteration-best policy selected by the Optimal Performance Search Agent (used as a warm-start for subsequent iterations), with shaded regions indicating 95% confidence intervals across random seeds. The lower panels show mean WIR (left) and RCR (right) per iteration.

because it lacks iterative feedback—once the initial reward function is generated, there is no mechanism to detect or correct misalignments. The ablation "w/o Warm-start" shows improvement across iterations but at a slower rate, because each iteration must re-explore the action space from scratch.

Fig. 8 quantifies each module's contribution. Without feedback, the Code Generation Agent commits to initial interpretations without verification, achieving a 16.3% failure rate—3× higher than the full model's 5.4%. This gap arises because LLM interpretations of ambiguous preferences often deviate from canonical targets, and the Result Analysis Agent is essential for detecting and correcting such deviations. Without warm-start, failure rate rises to 8.1% with slower convergence, particularly damaging for micro-level preferences where rule-satisfying regions are sparse.

The two modules are complementary: Result Analysis provides "what to improve" while Optimal Performance Search provides "where to start." Their combination yields a failure rate (5.4%) lower than either ablation alone (16.3% and 8.1%), confirming their synergy.

Table 6 reveals the computational implications. H3's per-timestep inference (5,486K tokens over 15 days) is prohibitively expensive; H2's per-episode calls accumulate 296K tokens. LA-UPAHEM achieves 57.3K tokens total by invoking LLMs only 15 times despite training

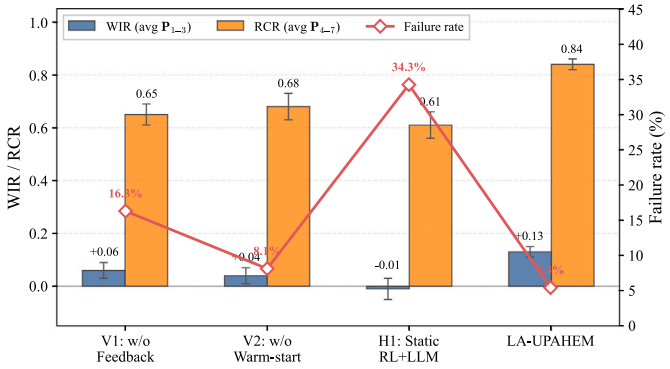


Fig. 8. Module-wise ablation. Grouped bars show average WIR ($\mathbf{P}_{1–3}$) and RCR ($\mathbf{P}_{4–7}$) per variant. Error bars denote across-seed standard deviation. The line (right axis) reports failure rate.

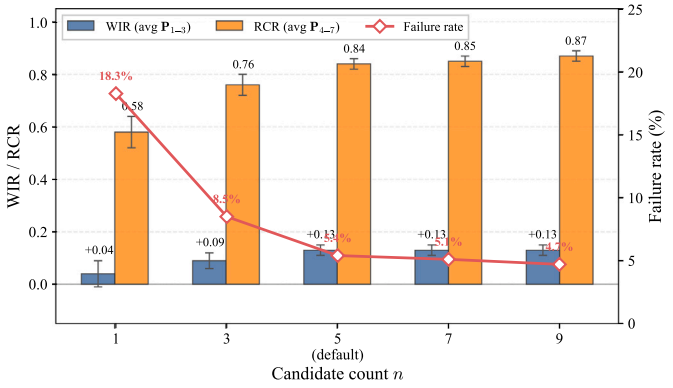


Fig. 9. Effect of the candidate count n on preference compliance (ι=5). Bars show WIR (avg $\mathbf{P}_{1–3}$) and RCR (avg $\mathbf{P}_{4–7}$); the line reports failure rate (right axis).

2250 episodes, with all overhead concentrated in training. The deployed policy operates independently without LLM inference, enabling reliable real-time control. Regarding wall-clock time, training a single 90-episode SAC run takes 6–8 minutes on our platform (Section 6.2); each outer iteration thus requires 30–40 minutes for n=5 candidates. LLM call latency varies with the deployment mode: cloud API calls depend on network conditions and provider queue times, whereas local inference incurs a fixed per-call cost. Overall, a complete ι=5 preference cycle finishes in approximately 2.5–3.5 hours end-to-end.

Regarding the two key hyper-parameters, Fig. 7 already demonstrates that iteration count ι exhibits diminishing returns: most WIR gains materialize within the first three iterations, and extending beyond ι=5 yields negligible improvement. Fig. 9 examines the complementary question of how the candidate count n affects performance. Generating a single candidate per iteration (n=1) substantially degrades WIR/RCR and raises the failure rate to 18.3%, because the framework loses exploration diversity and commits to a single reward formulation without alternatives. Increasing from n=5 to n=7 or n=9 yields negligible further gain while linearly increasing computational cost.

6.6. Robustness analysis

A practical HEMS must maintain consistent preference alignment under various perturbations. We evaluate robustness along four dimensions: exogenous noise, cross-environment generalization, evaluation weight sensitivity, and linguistic paraphrasing.

Table 7 shows how methods respond to Gaussian perturbations ($\beta \in \{0.00, 0.05, 0.10, 0.15\}$) on exogenous signals. Under comfort-prioritized $\mathbf{P}_3$, LA-UPAHEM maintains $\mathcal{L}_t \in [1.57, 2.01]$ across all noise levels—consistently the lowest. D2/D3 show $\mathcal{L}_t$ degrading by 52% (from 1.82 to 2.77), because they lack the state modification function $\mathcal{M}_{\mathbf{P}}$ which helps LA-UPAHEM recognize preference-relevant conditions under noisy inputs.

Cross-environment generalization is tested through random seeds that instantiate distinct residential configurations. Figs. 3 and 4 show LA-UPAHEM achieving not only higher mean WIR and RCR but also substantially lower variance than H1/H2—under $\mathbf{P}_1$, H1 varies by ±0.08 while LA-UPAHEM varies by less than ±0.03. This stability stems from two factors: (1) iterative refinement explores diverse reward structures rather than committing to a single potentially brittle formulation, and (2) successive performance feedback enables the LLM agent to identify which reward configurations yield consistent behavior across environments.

The canonical weights $\boldsymbol{\alpha}^\star$ used in our evaluation are produced by the LLM-based evaluator Φ_{eval} (Section 6.3). Because different users—or a different evaluator configuration—may assign somewhat different weights to the same natural-language preference, we verify that WIR rankings are robust to weight variation. For each of $\mathbf{P}_1$–$\mathbf{P}_3$, we scale the dominant weight by $\gamma \in \{0.8, 1.0, 1.2\}$ (±20%), proportionally redistribute the difference to the remaining weights, re-normalize, and recompute WIR from the existing KPI data (Table 5). Table 8 reports the results. At the canonical weighting (γ=1.0), method rankings are perfectly preserved (τ=1.00) for all three preferences. When the dominant weight is perturbed by ±20%, the stronger emphasis settings (γ=1.2) remain highly stable ($\tau \geq 0.87$), while the reduced emphasis settings (γ=0.8) lower τ to 0.73 due to minor rank swaps among middle-tier methods whose WIR values cluster near zero. Crucially, LA-UPAHEM retains the top position across all nine $(\mathbf{P}, \gamma)$ combinations. These results confirm that evaluation conclusions are robust to reasonable weight variation.

Finally, we test whether LA-UPAHEM's performance depends on the specific wording of user preferences. For one representative preference per category (macro $\mathbf{P}_1$, micro $\mathbf{P}_4$, compositional $\mathbf{P}_7$), we construct three semantically equivalent paraphrases by varying register, sentence structure, and verbosity (Table 9). Across all twelve paraphrased inputs (ι=5, n=5, DeepSeek V3, 10 seeds), WIR and RCR exhibit standard deviations of at most 0.04, and failure vary by 1.5–2.2 percentage points. $\mathbf{P}_1$ reports no RCR because it contains no micro-level rules. The negative WIR values for $\mathbf{P}_4$ and $\mathbf{P}_7$ reflect the inherent macro-level KPI cost of micro-rule compliance, consistent with our metric design (Section 6.3). This stability arises because the iterative feedback mechanism evaluates KPI outcomes rather than linguistic similarity: even when the initial

Table 6
Computational cost over a 15-day preference cycle (DeepSeek V3, SAC backend). Training occurs once per preference change; inference accumulates over 15 days (288 × 15 = 4320 timesteps).

	Training Phase			Inference Phase		
Method	Episodes	LLM Calls	Tokens/Call (K)	LLM Calls	Tokens/Call (K)	Total Tokens (K)
H1: Static RL + LLM	90	1	2.15	0	–	2.15
H2: LLM-Reward	90	90	3.29	0	–	296.10
H3: LLM-Action Filter	0	0	–	4320	1.27	5,486.40
LA-UPAHEM (Ours)	2250	15	3.82	0	–	57.30

Table 7
Raw KPI values under different noise intensities β for preference $\mathbf{P}_3$ (comfort-prioritized). D2/D3 are merged because $\mathbf{P}_3$ contains no micro-level rules (cf. Table 5); values at $\beta > 0$ are averaged. $\mathcal{L}_e$ values scaled ×10.

Method	$\beta = 0.00$			$\beta = 0.05$			$\beta = 0.10$			$\beta = 0.15$		
	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$	$\mathcal{L}_c$	$\mathcal{L}_t$	$10\mathcal{L}_e$
D1: Fixed-Reward DRL	8.10	2.49	1.49	8.49	2.73	1.51	8.95	3.00	1.53	9.42	3.26	1.55
D2/D3: Oracle DRL	9.38	1.82	1.74	9.84	2.13	1.77	10.34	2.47	1.80	10.81	2.77	1.83
H1: Static RL + LLM	8.91	2.24	1.87	9.30	2.55	1.89	9.78	2.84	1.92	10.22	3.14	1.95
H2: LLM-Reward	9.41	2.03	1.78	9.83	2.29	1.80	10.31	2.60	1.83	10.79	2.88	1.86
H3: LLM-Action Filter	9.45	2.12	1.86	9.89	2.40	1.88	10.40	2.73	1.91	10.88	3.02	1.94
LA-UPAHEM	9.71	1.57	1.82	9.95	1.69	1.83	10.20	1.85	1.85	10.43	2.01	1.87

Table 8
Sensitivity of WIR rankings to canonical weight perturbation (γ: scaling factor on the dominant $\alpha_k^\star$; remaining weights proportionally redistributed). D2/D3 are merged because $\mathbf{P}_1$–$\mathbf{P}_3$ contain no micro-level rules, making their policies identical (cf. Table 5). τ: Kendall's rank correlation with the canonical ranking.

Pref.	γ	D1	D2/D3	H1	H2	H3	Ours	τ
$\mathbf{P}_1$	0.8	0.00	−0.08	−0.14	−0.12	−0.11	**0.00**	0.73
	1.0	0.00	−0.03	−0.11	−0.06	−0.06	**0.04**	1.00
	1.2	0.00	0.01	−0.08	0.00	−0.01	**0.09**	0.87
$\mathbf{P}_2$	0.8	0.00	0.02	−0.02	0.02	−0.05	**0.04**	0.73
	1.0	0.00	0.11	0.06	0.12	0.02	**0.15**	1.00
	1.2	0.00	0.19	0.13	0.21	0.09	**0.27**	1.00
$\mathbf{P}_3$	0.8	0.00	0.08	−0.02	0.03	−0.01	**0.11**	0.73
	1.0	0.00	0.14	0.02	0.08	0.04	**0.20**	1.00
	1.2	0.00	0.20	0.06	0.13	0.09	**0.28**	1.00

Table 9
Paraphrase robustness. Each preference is tested with the original phrasing and three semantically equivalent paraphrases (informal, restructured, terse/concise). All runs use DeepSeek V3 with ι=5, n=5, averaged over 10 seeds.

Pref.	Phrasing	WIR	RCR	Failure Rate
$\mathbf{P}_1$	Original	0.04	–	5.2%
	Informal: "*Keeping my bill low matters more than perfect temperature.*"	0.01	–	7.6%
	Restructured: "*I don't mind discomfort, as long as I save on energy.*"	0.06	–	5.4%
	Terse: "*Cut power costs even if less comfortable.*"	0.08	–	4.0%
	Mean ± Std	0.05±0.03	–	5.6±1.5%
$\mathbf{P}_4$	Original	−0.25	0.84	6.2%
	Emotional: "*When I'm home, don't charge the battery—it feels unsafe.*"	−0.30	0.78	8.6%
	Restructured: "*Only charge the battery when nobody is home, for safety.*"	−0.23	0.87	4.4%
	Indirect: "*Avoid battery charging during occupied hours.*"	−0.32	0.80	9.2%
	Mean ± Std	−0.28±0.04	0.82±0.04	7.1±2.2%
$\mathbf{P}_7$	Original	−0.29	0.81	5.8%
	Concise: "*Don't heat below setpoint—I'll layer up. Save electricity.*"	−0.35	0.75	8.8%
	Formal: "*If room temperature is below target, leave AC off to cut costs.*"	−0.27	0.83	4.6%
	Rephrased: "*I'd rather tolerate cold than run the AC below set temperature.*"	−0.36	0.76	8.0%
	Mean ± Std	−0.32±0.04	0.79±0.04	6.8±1.9%

reward code differs between phrasings, the Result Analysis Agent identifies the same behavioral deficiencies and guides convergence toward similar final policies.

6.7. Generality across algorithms and LLMs

To verify whether LA-UPAHEM generalizes beyond the default configuration, we evaluate its compatibility with alternative DRL algorithms and LLM backbones.

Table 10 examines DRL algorithm compatibility. Among single-agent algorithms, SAC demonstrates the best sample efficiency, benefiting from entropy regularization and off-policy replay. PPO shows slower initial learning but strong final performance, as this on-policy algorithm requires more data to converge. TD3 achieves moderate performance with stable training. DDPG fails persistently, with failure rates exceeding 72% even at 270 episodes, because its deterministic policy and lack of entropy regularization make it prone to converging on degenerate solutions when the reward function changes across iterations.

For multi-agent DRL (MADRL), we adopt the Centralized Training with Decentralized Execution (CTDE) paradigm with separate agents for AC and BESS. MASAC achieves performance close to single-agent SAC, demonstrating compatibility with CTDE frameworks. MADDPG substantially outperforms its single-agent counterpart DDPG, likely because the decomposed per-device action spaces reduce each agent's decision complexity. Overall, entropy-regularized algorithms (SAC, MASAC, PPO) offer the best reliability.

Table 11 evaluates LLM backbone compatibility across three languages. DeepSeek V3 achieves the highest English performance but shows slight degradation for Chinese/Spanish; the "Best Iter." column indicates that additional iterations compensate for non-English interpretation gaps. Commercial APIs (GPT-4, Claude) provide the most consistent cross-language results with minimal degradation and low failure rates. LLaMA-3 70B performs acceptably in English but degrades significantly for Chinese, suggesting domain-specific fine-tuning may be needed for multilingual support.

The performance gap between LLM backbones is driven primarily by code generation quality rather than preference understanding. The percentage of generated code passing the validation pipeline (Appendix E) before retry drops from 93–96% for commercial APIs to 68–78% for LLaMA-3 70B, directly explaining the failure-rate gap (3–5% vs. 15–24%): the retry mechanism (N=50) can recover from occasional malformed outputs but not from systematically low code quality.

To assess the feasibility of fully on-premise deployment for privacy-sensitive scenarios (Section 5.5), we also evaluate LLaMA-3 8B. Its failure rates reach 48–68% across languages, confirming that current 8B-class models cannot reliably generate valid reward code under zero-shot prompting. The "Best Iter." of 5.0 across all languages indicates that 8B models consistently exhaust the maximum iterations without converging. For practitioners, commercial APIs offer the best reliability

Table 10
Generality across DRL algorithms (WIR averaged over $\mathbf{P}_{1:3}$, RCR averaged over $\mathbf{P}_{4:7}$) with varying training budgets. Single-agent algorithms (top) are compared with multi-agent DRL (bottom).

Algorithm	30 Episodes			90 Episodes			270 Episodes		
	WIR	RCR	Failure Rate	WIR	RCR	Failure Rate	WIR	RCR	Failure Rate
Single-agent DRL									
SAC [41]	0.08	0.68	12.4%	0.13	0.84	5.4%	0.14	0.86	4.2%
TD3 [44]	0.05	0.61	18.3%	0.09	0.77	10.1%	0.11	0.80	7.8%
PPO [45]	0.02	0.51	22.5%	0.11	0.79	8.3%	0.14	0.85	5.2%
DDPG [46]	−0.05	0.28	91.7%	−0.01	0.38	85.4%	0.03	0.51	72.3%
Multi-agent DRL (MADRL)									
MASAC	0.04	0.54	21.2%	0.10	0.78	8.7%	0.13	0.82	5.8%
MADDPG [47]	−0.01	0.38	35.4%	0.05	0.64	17.6%	0.08	0.71	12.8%

Table 11
Generality across LLM backbones and preference languages (WIR averaged over $\mathbf{P}_{1:3}$, RCR averaged over $\mathbf{P}_{4:7}$). Best Iter. = iteration at which the optimal $(\mathcal{R}_{\mathbf{P}}, \mathcal{M}_{\mathbf{P}})$ pair was found.

LLM Model	Language	WIR	RCR	Failure Rate	Best Iter.
DeepSeek V3 (671B) [42]	English	0.13	0.84	5.4%	2.8
	Chinese	0.11	0.80	7.8%	3.4
	Spanish	0.11	0.81	6.5%	3.2
GPT-4 API [36]	English	0.12	0.82	3.2%	2.5
	Chinese	0.11	0.80	4.4%	2.8
	Spanish	0.11	0.81	3.8%	2.7
Claude Sonnet 4.0 API [48]	English	0.11	0.81	3.4%	2.6
	Chinese	0.10	0.79	5.2%	3.0
	Spanish	0.11	0.80	4.6%	2.9
LLaMA-3 70B [49]	English	0.09	0.78	14.8%	3.5
	Chinese	0.06	0.70	23.6%	4.5
	Spanish	0.07	0.74	19.5%	4.1
LLaMA-3 8B [49]	English	0.04	0.53	48.2%	5.0
	Chinese	−0.01	0.37	67.5%	5.0
	Spanish	0.01	0.43	59.8%	5.0

when multilingual support is needed, while DeepSeek V3 is preferable for English-only deployments where cost efficiency matters.

Overall, LA-UPAHEM generalizes well across DRL backends (SAC, PPO, TD3, MASAC) and LLM providers (DeepSeek, GPT-4, Claude), with entropy-regularized algorithms and commercial LLMs offering the best reliability for practical deployment.

7. Conclusion and future work

This paper presents LA-UPAHEM, an iterative LLM-augmented framework that translates natural language preferences into DRL-based home energy control. By combining three specialized agents in a closed-loop refinement process, the framework enables residential control strategies to continuously adapt to evolving user preferences without requiring manual reward engineering.

Experiments on real residential energy datasets demonstrate that LA-UPAHEM consistently outperforms classical DRL, oracle-assisted, and existing RL + LLM baselines in both macro-level KPI alignment (cost, comfort, emissions) and micro-level rule compliance. The iterative refinement mechanism reduces failure rates from 34.3% (static one-shot) to 5.4%, while the decoupled architecture ensures that deployed policies operate independently without LLM inference overhead. Module-wise ablation confirms the complementary roles of the feedback and warm-start mechanisms, and robustness experiments show stable performance under environmental noise, evaluation weight perturbation, and linguistic paraphrasing. Generalization experiments further validate compatibility across multiple DRL algorithms (SAC, PPO, TD3, MASAC), LLM backbones, and preference languages.

A key limitation is that the current framework operates at episode granularity: when a user's preference changes, a new iterative optimization cycle is triggered (~2.5–3.5 hours). Intra-episode adaptation—for example, a user abruptly changing comfort priority mid-day—is outside the current scope. Potential mitigations include maintaining a library of pre-optimized policies indexed by preference type for rapid switching, or developing online reward adaptation mechanisms that adjust the deployed policy without full retraining.

Future work includes integrating multimodal preference cues (e.g., occupant behavior patterns, wearable sensor data), enabling lightweight local deployment through few-shot prompting or task-specific fine-tuning to improve small-model code generation quality (Section 5.5), and extending to community-scale coordination across multiple households.

CRediT authorship contribution statement

Xiao Du: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Fengji Luo:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Data curation, Conceptualization. **Juntao Hu:** Writing – review & editing, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Wei Zhou:** Writing – review & editing, Supervision, Software, Resources, Project administration, Investigation, Conceptualization. **Junhao Wen:** Writing – review & editing, Supervision, Software, Resources, Project administration, Conceptualization.

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.

Appendix A. Household environment parameters

Table 12 presents the complete parameter configuration for the household environment used in our experiments.

Table 12
Modeling parameters of the household environment.

Parameter	Notation	Value
Thermal Dynamics		
Specific heat / density of air	c^{air}, ρ^{air}	3×10^{-4} kWh/(kg· °C), 1.30 kg/m^3
House volume	V^{hou}	420 m^3
Thermal conductivity	λ	9.3×10^{-4} kW/(m^2· °C)
Envelope surface / window area	A^{suf}, A^{fen}	266 m^2, 16 m^2
Air change rate / SHGC	ACH, SHGC	0.7 h^{-1}, 0.30
Air Conditioner (AC)		
Rated / standby power	$P^{\text{a}}_{\text{rat}}$, $P^{\text{a}}_{\text{sta}}$	1.5 kW, 0.010 kW
Efficiency / rated COP	η^{a}, cop_{rat}	0.08, 3.51
Battery Energy Storage System (BESS)		
Rated power / capacity	$P^{\text{b}}_{\text{rat}}$, E_{rat}	3.0 kW, 50 kWh
Efficiency / self-discharge rate	η^{b}, SD	0.98, 0.005 h^{-1}
Photovoltaic (PV) System		
Rated output / efficiency	$P^{\text{PV}}_{\text{rat}}$, η^{PV}	0.28 kW/m^2, 0.20
Min. irradiance / panel area	I_{min}, A^{pan}	0.01 kW/m^2, 10 m^2
Stochastic Occupancy Model		
Probability of going out (work/rest)	$p^{\text{out}}_{\text{work}}$, $p^{\text{out}}_{\text{rest}}$	0.90, 0.45
Mean departure time (work/rest)	$\mu^{\text{dep}}_{\text{work}}$, $\mu^{\text{dep}}_{\text{rest}}$	9:00, 11:00
Std. dev. of departure (work/rest)	$\sigma^{\text{dep}}_{\text{work}}$, $\sigma^{\text{dep}}_{\text{rest}}$	1.0 h, 2.0 h
Mean arrival time (work/rest)	$\mu^{\text{arr}}_{\text{work}}$, $\mu^{\text{arr}}_{\text{rest}}$	19:00, 17:00
Std. dev. of arrival (work/rest)	$\sigma^{\text{arr}}_{\text{work}}$, $\sigma^{\text{arr}}_{\text{rest}}$	1.0 h, 2.0 h
Comfort Settings		
Temperature setpoint / deviation band	T^{set}, T^{band}	19 °C, 2 °C

Appendix B. Data provenance and preprocessing

Table 13 summarizes the data sources, fields, and preprocessing steps used to construct the simulation environment.

Table 13
Data provenance. All streams share 1-minute native resolution and are downsampled to 5-minute intervals for the control MDP. Missing weather values are imputed by backward-fill then forward-fill.

Data stream	Source	Fields used	Notes
Household load	Pecan Street [38], Apt. #3938, San Diego, CA	Equipment electric power (kW)	2016; 120-day summer window used
Weather	NOAA BSRN, Trinidad Head (THD) station, CA [39]	Dry-bulb temperature (°C), global solar irradiance (W/m^2)	Missing: solar 0.32%, temp. 5.72%
Electricity price	CityLearn v2 [40]	Price ($/kWh)	Time-of-use (TOU) schedule
Carbon intensity	CityLearn v2 [40]	Grid CO_2 intensity (kg/kWh)	Hourly profile, 2016
Occupancy	Stochastic model	Binary (home/away)	See Table 12

Appendix C. Prompt templates

This appendix provides the complete prompt templates used by the three LLM agents in LA-UPAHEM. The functional roles and inter-agent mechanisms are described in Section 5; here we present the actual prompts for reproducibility.

C.1. Code generation agent prompt

System Prompt:

```
You are an expert in reinforcement learning for home energy management systems. You will
generate BOTH a state modifier and a reward function as a pair.

The state modifier processes the original state first, then the reward function computes reward
on the modified state. They must be designed together for consistency.
```

User Prompt Structure:

Table 14 summarizes the structured sections of the user prompt, each dynamically populated from the task-specific context **C** and user preferences **P**.

Table 14
Code generation Agent user prompt composition.

Section	Source	Content
Task Overview	$\mathbf{c}_{\text{des}}$	HEMS control task, controllable devices (AC, BESS), action space (control signal, desired state, desired power), grid constraints. *E.g., "The AC action is a continuous signal in* $[-1, 1]$*: positive values activate cooling toward a desired temperature."*
KPI Definitions	$\mathbf{c}_{\text{kpi}}$	KPI metrics with names, units, optimization directions. *E.g., "*`electricity_cost` *($/day, minimize): total daily grid import cost;* `thermal_discomfort` *(°C·h, minimize): cumulative deviation from set-point."*
User Preference	$\mathbf{P}$	Natural language preference, preference name, focus KPI areas
Observation Space	$\mathbf{c}_{\text{des}}$	33 state variables (+5 optional), each with index, type, range, unit. *E.g., "idx 6:* `indoor_temp`*, float, [15, 35] °C, current indoor air temperature; idx 18:* `battery_soc`*, float, [0, 1], battery state of charge."*
State Dictionary	$\mathbf{c}_{\text{des}}$	Access pattern: `state['name_history'][-1]` for current value, `state['name_history']` for full history
Reference Code	$\mathbf{c}_{\text{rref}}, \mathbf{c}_{\text{mref}}$	Complete `StateModifier` and `Reward` reference implementations
Output Format	–	Two Python code blocks: `StateModifier` first, `Reward` second
Additional for update mode (iterations $i > 1$*):*		
Previous Experience	$\mathcal{D}$	Last m iterations' code, KPIs, and analysis feedback

The complete user prompt is assembled by concatenating the sections listed in Table 14 in order, followed by the core generation instruction below:

```
Generate both classes following this workflow:
1. StateModifier.modify_state(original_state) -> modified_state
2. Reward.compute_reward(modified_state) -> reward

Design them as a pair:
- If state_modifier adds new features, reward should use them
- If state_modifier doesn't need modifications, create an IdentityStateModifier
- Ensure both classes work together consistently
```

In update mode (iterations $i > 1$), the previous iteration experience from the memory pool $\mathcal{D}$ (Table 14, last row) is appended, followed by an additional instruction:

```
Based on previous results, generate improved paired state_modifier and reward functions.
```

C.2. Result analysis agent prompt

System Prompt:

```
You are an expert in analyzing reinforcement learning training results for home energy
management systems.
```

User Prompt Input: The task-specific context (Table 14, rows 1–5), plus training results for all n candidates (state modifier code, reward code, trajectories, KPI metrics).
Analysis Instructions (verbatim):
In the following prompts, `{P}` and `{focus_areas}` are runtime-populated placeholders for the user's natural language preference and focus KPI areas, respectively.

```
Analyze the results and provide suggestions:
1. Evaluate how well each reward function aligns with the user preference: "{P}"
2. Identify what works well and what doesn't
3. Focus on the preference areas: {focus_areas}
4. Suggest specific improvements for the next iteration

Keep your analysis concise and actionable.
```

C.3. Optimal performance search agent prompt

System Prompt:

```
You are an expert in evaluating reinforcement learning results for home energy management.
```

User Prompt Input: The task-specific context, plus KPI results and analysis from the search range. Reward code is excluded to avoid biasing selection toward code style.

Selection Instructions (verbatim):

```
Select the best reward function based on:
1. Alignment with user preference: "{P}"
2. Overall KPI performance
3. Focus on preference areas: {focus_areas}
4. Agent behavior quality

Provide your answer in this exact format (use INTEGER numbers only):
- Index: (iteration_number, sample_number)
- Reason: Your explanation
```

Appendix D. Reference code templates

The following Python implementations correspond to the state modification template $\mathbf{c}_{\text{mref}}$ (Eq. 11) and the reward function template $\mathbf{c}_{\text{rref}}$ (Eq. 12) described in Section 5.1. These are provided to the Code Generation Agent as structural references; the generated code may deviate substantially from these templates.

```
# === State Modification Template (c_mref) ===
class StateModifier(BaseStateModifier):
  def modify_state(self, original_state):
    modified_state = original_state.copy()
    # Electricity cost intensity
    cost = abs(original_state['home_electrical_cost_history'][-1])
    kappa = original_state['electricity_pricing_history'][-1]
    modified_state['cost_intensity_history'] = [-cost * kappa]
    # Thermal discomfort degree
    temp_in = original_state['indoor_temperature_history'][-1]
    temp_set = original_state['target_temperature_history'][-1]
    T_band = 2.0
    modified_state['discomfort_history'] = [
      min(0, T_band - abs(temp_in - temp_set))]
    # Carbon emission intensity
    mu = original_state['carbon_intensity_history'][-1]
    modified_state['carbon_intensity_history'] = [-cost * mu]
    return modified_state
```

```
# === Reward Function Template (c_rref) ===
import math
class Reward(BaseReward):
  def compute_reward(self, state):
    cost_intensity = state['cost_intensity_history'][-1]
    discomfort = state['discomfort_history'][-1]
    carbon = state['carbon_intensity_history'][-1]
    occ = state['occupancy_history'][-1]
    # Bounded rewards via exponential transform
    r_cost = math.exp(cost_intensity)
    r_comfort = occ * math.exp(discomfort)
    r_carbon = math.exp(carbon)
    # Equal-weight baseline
    reward = 0.33 * r_cost + 0.34 * r_comfort + 0.33 * r_carbon
    return reward
```

Appendix E. Code generation safety measures

This appendix details the five-layer validation pipeline introduced in Section 5.4. Each layer targets a distinct failure mode:

1. *Interface constraints*: all generated code must inherit from abstract base classes (`BaseReward` with method `compute_reward(state)` returning `float`; `BaseStateModifier` with method `modify_state(state)` returning `dict`), ensuring structural compatibility with the environment wrapper.
2. *Syntactic validation*: each LLM response must contain at least two Python code blocks, which are imported as modules and checked for the presence of required class names.
3. *Runtime testing*: a test HEMS environment is instantiated with the generated classes, and five random action steps are executed to verify that the state modifier processes states without exceptions and the reward function returns valid numerical values.
4. *Retry mechanism*: up to $N_{\text{try}} = 50$ generation attempts are allowed per sample, with automatic rejection and regeneration upon any validation failure. In practice, commercial APIs (GPT-4, Claude, DeepSeek V3) typically pass within 1–3 attempts, while LLaMA-3 70B requires 5–8 attempts (Section 6.7).
5. *State space documentation*: the prompt template explicitly specifies all 33 observation features with their indices, types, ranges, and units (Appendix C.1, Table 14), constraining the generated code to access only documented state variables.

Data availability

Data will be made available on request.

References

[1] Rokonuzzaman M, Rahman S, Hannan MA, Mishu MK, Tan W-S, Rahman KS, Pasupuleti J, Amin N. Levenberg-marquardt algorithm-based solar pv energy integrated internet of home energy management system. Appl Energy 2025;378.
[2] Tuomela S, de Castro Tomé M, Iivari N, Svento R. Impacts of home energy management systems on electricity consumption. Appl Energy 2021;299:117310.
[3] Chen X, Qu G, Tang Y, Low S, Li N. Reinforcement learning for selective key applications in power systems: recent advances and future challenges. IEEE Trans Smart Grid 2022;13:2935–58.
[4] Xiong S, Liu D, Chen Y, Zhang Y, Cai X. A deep reinforcement learning approach based energy management strategy for home energy system considering the time-of-use price and real-time control of energy storage system. Energy Rep 2024;11:3501–8.
[5] Ren K, Liu J, Wu Z, Liu X, Nie Y, Xu H. A data-driven drl-based home energy management system optimization framework considering uncertain household parameters. Appl Energy 2024;355:122258.
[6] Yang Y, Hao J, Zheng Y, Yu C. Large-scale home energy management using entropy-based collective multiagent deep reinforcement learning framework. In: IJCAI; 2019. p. 630–6.
[7] Liu ZE, Zhou Q, Li Y, Shuai S, Xu H. Safe deep reinforcement learning-based constrained optimal control scheme for HEV energy management. IEEE Trans Transp Electrif 2023;9:4278–93.
[8] Gokhale G, Claessens B, Develder C. Explainable home energy management systems based on reinforcement learning using differentiable decision trees. In: Proceedings of the 15th ACM international conference on future and sustainable energy systems; 2024. p. 687–91.
[9] Gao J, Wang W, Nikseresht F, Govinda Rajan V, Campbell B. PFDRL: personalized federated deep reinforcement learning for residential energy management. In: Proceedings of the 52nd international conference on parallel processing; 2023. p. 402–11.
[10] Sumayli M, Anubi OM. Integration of multi-mode preference into home energy management system using deep reinforcement learning. arXiv:2505.01332 [arXiv preprint] 2025.
[11] Saroha P, Singh G, Lilhore UK, Simaiya S, Khan M, Alroobaea R, Alsafyani M, Alsufyani H. Dynamic appliance scheduling and energy management in smart homes using adaptive reinforcement learning techniques. Sci Rep 2025;15:24594.
[12] Parrish B, Heptonstall P, Gross R, Sovacool BK. A systematic review of motivations, enablers and barriers for consumer engagement with residential demand response. Energy Policy 2020;138:111221.
[13] Chen S-J, Chiu W-Y, Liu W-J. User preference-based demand response for smart home energy management using multiobjective reinforcement learning. IEEE Access 2021;9:161627–37.
[14] Kaplar A, Vidaković M, Vidaković J, Slivka J. House energy management system for balancing electricity costs and residential comfort based on deep reinforcement learning. Acta Polyt Hung 2024;21:289–308.
[15] Yao L, Liu P-Y, Teo JC. Hierarchical multi-agent deep reinforcement learning with adjustable hierarchy for home energy management systems. Energy Build 2025;331:115391.
[16] Almeida R, Georgieva P, Martins N. Heating setpoint recommendation strategy for thermal comfort and energy consumption optimization. Energy Build 2023;296:113406.
[17] Liu Y, Ma J, Xing X, Liu X, Wang W. A home energy management system incorporating data-driven uncertainty-aware user preference. Appl Energy 2022;326:119911.
[18] Nagy Z, Henze G, Dey S, Arroyo J, Helsen L, Zhang X, Chen B, Amasyali K, Kurte K, Zamzam A, et al. Ten questions concerning reinforcement learning for building energy management. Build Environ 2023;241:110435.
[19] Mahmood S, Sun H, Ali Alhussan A, Iqbal A, El-Kenawy E-SM. Active learning-based machine learning approach for enhancing environmental sustainability in green building energy consumption. Sci Rep 2024;14:19894.
[20] Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. Adv Neural Inf Process Syst 2022;35:22199–213.
[21] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, et al. Training language models to follow instructions with human feedback. Adv Neural Inf Process Syst 2022;35:27730–44.
[22] Sun G, Xie W, Niyato D, Mei F, Kang J, Du H, Mao S. Generative AI for deep reinforcement learning: framework, analysis, and use cases. IEEE Wirel Commun 2025;32(3):186–195.
[23] Xu F, Hao Q, Zong Z, Wang J, Zhang Y, Wang J, Lan X, Gong J, Ouyang T, Meng F, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. arXiv:2501.09686 [arXiv preprint] 2025.
[24] Qiu D, Chen Y, Niu M, Feng F, Zhao Y, Huang J, Xu M, Yin D. Embodied executable policy learning with language-based scene summarization. In: Proceedings of the 2024 conference of the north American chapter of the association for computational linguistics: human language technologies (NAACL); 2024.
[25] Agashe S, Fan Y, Wang XE. Evaluating multi-agent coordination abilities in large language models. arXiv:2310.03903 [arXiv preprint] 2023.
[26] Carta T, Romac C, Wolf T, Lamprier S, Sigaud O, Oudeyer P-Y. Grounding large language models in interactive environments with online reinforcement learning. In: Proceedings of the 40th international conference on machine learning (ICML); 2023. p. 3676–713.
[27] Tang X, Liu F, Xu D, Jiang J, Tang Q, Wang B, Wu Q, Chen CLP. Llm-assisted reinforcement learning: leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment. IEEE Trans Consum Electron 2025;71(2):5631–5644.
[28] Yuan H, Zhang C, Wang H, Xie F, Cai P, Dong H, Lu Z. Skill reinforcement learning and planning for open-world long-horizon tasks. arXiv:2303.16563 [arXiv preprint] 2023.
[29] Hu H, Sadigh D. Language instructed reinforcement learning for human-ai coordination. In: International conference on machine learning. PMLR; 2023. p. 13584–98.
[30] Karine K, Marlin BM. Combining llm decision and rl action selection to improve rl policy for adaptive interventions. arXiv:2501.06980 [arXiv preprint] 2025.
[31] Kwon M, Michael S. Reward design with language models. In: International conference on learning representations (ICLR); 2023.
[32] Ma YJ, Liang W, Wang G, Huang D-A, Bastani O, Jayaraman D, Zhu Y, Fan L, Anandkumar A. Eureka: human-level reward design via coding large language models. In: International conference on learning representations (ICLR); 2024.
[33] Zhang S, Zheng S, Ke S, Liu Z, Jin W, Yuan J, Yang Y, Yang H, Wang Z. How can llm guide rl? a value-based approach. arXiv:2402.16181 [arXiv preprint] 2024.
[34] Klissarov M, D'Oro P, Sodhani S, Raileanu R, Bacon P-L, Vincent P, Zhang A, Henaff M. Motif: intrinsic motivation from artificial intelligence feedback. In: International conference on learning representations (ICLR); 2024.
[35] Wang B, Qu Y, Jiang Y, Shao J, Liu C, Yang W, Ji X. Llm-empowered state representation for reinforcement learning. In: Proceedings of the 41st international conference on machine learning (ICML); 2024.
[36] Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Aleman FL, Almeida D, Altenschmidt J, Altman S, Anadkat S, et al. Gpt-4 technical report. arXiv:2303.08774 [arXiv preprint] 2023.
[37] Valmeekam K, Sreedharan S, Marquez M, Olmo A, Kambhampati S. On the planning abilities of large language models: a critical investigation. In: Advances in neural information processing systems (NeurIPS); 2023.
[38] Pecan Street Inc. Pecan street database; 2020. http://www.pecanstreet.org/[Accessed: June 2024].
[39] National Oceanic and Atmospheric Administration (NOAA). NOAA data, Online:; 2020. https://www.ncdc.noaa.gov/[Accessed: June 2024].

[40] Nweye K, Kaspar K, Buscemi G, Fonseca T, Pinto G, Ghose D, Duddukuru S, Pratapa P, Li H, Mohammadi J, Lino Ferreira L, Hong T, Ouf M, Capozzoli A, Nagy Z. Citylearn v2: energy-flexible, resilient, occupant-centric, and carbon-aware management of grid-interactive communities. J Build Perform Simul 2024;18(1):17–38.
[41] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861–70.
[42] Guo D, Yang D, Zhang H, Song J, Zhang R, Xu R, Zhu Q, Ma S, Wang P, Bi X, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv:2501.12948 [arXiv preprint] 2025.
[43] Dubois Y, Galambosi B, Liang P, Hashimoto TB. Length-controlled alpacaeval: A simple way to debias automatic evaluators. arXiv:2404.04475 [arXiv preprint] 2024.
[44] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR; 2018. p. 1587–96.
[45] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347 [arXiv preprint] 2017.
[46] Duan Y, Chen X, Houthooft R, Schulman J, Abbeel P. Benchmarking deep reinforcement learning for continuous control. In: International conference on machine learning. PMLR; 2016. p. 1329–38.
[47] Lowe R, Wu YI, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in neural information processing systems, vol. 30. 2017. p. 6379–90.
[48] Anthropic. Claude opus 4 and claude sonnet 4 system card; 2025. https://www.anthropic.com/research/claude-4-system-card[accessed: March 2026].
[49] Grattafiori A, Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, Mathur A, Schelten A, Yang A, et al. The llama 3 herd of models. arXiv:2407.21783 [arXiv preprint] 2024.