Multi-Agent Large Language Model Framework for Window Optimization

Bridging human design intent and data-driven environmental performance

Reza Taghavifard[1] and Negin Cheraghi[2]
[1]*University of Montreal*
[2]*Iran University of Science and Technology*
[1]*reza.taghavifard@umontreal.ca, 0009-0003-5267-2546*
[2]*negincheraqi@gmail.com, 0009-0006-6902-7422*

Abstract. Optimizing window placement and dimensions is crucial to balancing daylight access, glare control, and energy efficiency. However, reconciling multiple, often conflicting objectives remains challenging, particularly when qualitative user intent is excluded from the optimization loop. This paper presents a multi-agent Large Language Model (LLM) framework that integrates natural language reasoning, computer vision, and environmental simulation to generate context-aware, energy-efficient window configurations. The system comprises three interdependent agents: The first agent interacts with users and processes floor plan drawings via a YOLOv8-based detection model. The second agent performs daylight and glare analyses using Honeybee and Ladybug in Python, feeds result to the first agent, and defines objective functions dynamically based on user needs. The third agent executes NSGA-II multi-objective optimization, periodically analyzing the evolving Pareto front and reporting key trade-offs to the first agent, which can update objectives and constraints based on user feedback and trigger early stopping once satisfactory solutions are reached. By embedding user priorities and data-driven insights, the framework interprets and selects optimal configurations from the Pareto front, generating personalized, context-aware recommendations. Results confirm the system's effectiveness in harmonizing technical performance with subjective comfort, reducing decision complexity and advancing intelligent, user-adaptive window optimization.

Keywords. Window optimization, Multi-agent framework, Large Language Models (LLMs), Multi-objective optimization

1. Introduction

As global energy demands grow, improving building performance has become essential for sustainability and emission reduction (Trząski & Rucińska, 2025). Windows critically influence daylight and solar gain(Chi, 2021). factors such as window-to-wall ratio (WWR) and orientation strongly impact thermal and visual

HUMANISTIC COMPUTATION AND INTELLIGENCE, Proceedings of the 31st International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2026, Volume 3, 397-406.

comfort, and poor designs may cause glare or uneven daylight(Liu et al., 2021). Energy optimization in buildings requires balancing often conflicting criteria, including energy efficiency, daylight quality, and visual comfort. This study employs NSGA-II to handle trade-offs among these objectives, while extending its applicability beyond expert-driven workflows. (Zhao & Du, 2020). However, such methods still rely heavily on structured numerical inputs and expert supervision, limiting their accessibility for non-expert users and placing the burden of qualitative interpretation on domain specialists(Diakaki et al., 2008).

Large Language Models (LLMs), capable of translating unstructured natural language into structured, quantifiable parameters can bridge this communication gap. (Jiang et al., 2024; Zhang et al., 2024; Zhang & Chen, 2025). By aligning user intent with performance targets such as daylight autonomy and glare probability, LLMs enable interactive, human-in-the-loop optimization workflows that extend beyond rule-based automation toward intent-driven performance exploration.(Bagasi et al., 2025). Integrated with multi-agent systems, they act as mediators between user goals, regulations, and building physics, rather than operating as isolated language interfaces (Hou et al., 2024; Postle & Salingaros, 2025).

This study focuses on two interdependent performance parameters, daylighting and glare, as key drivers of window-related energy performance(Konstantzos et al., 2015). Building on LLM-based automation and multi-objective optimization (MOO), a multi-agent LLM framework is proposed to translate qualitative design intent into quantitative simulation for window optimization, addressing a gap between user-driven reasoning and performance-based modeling that is often unaddressed in conventional optimization or LLM-only approaches. The framework autonomously derives window

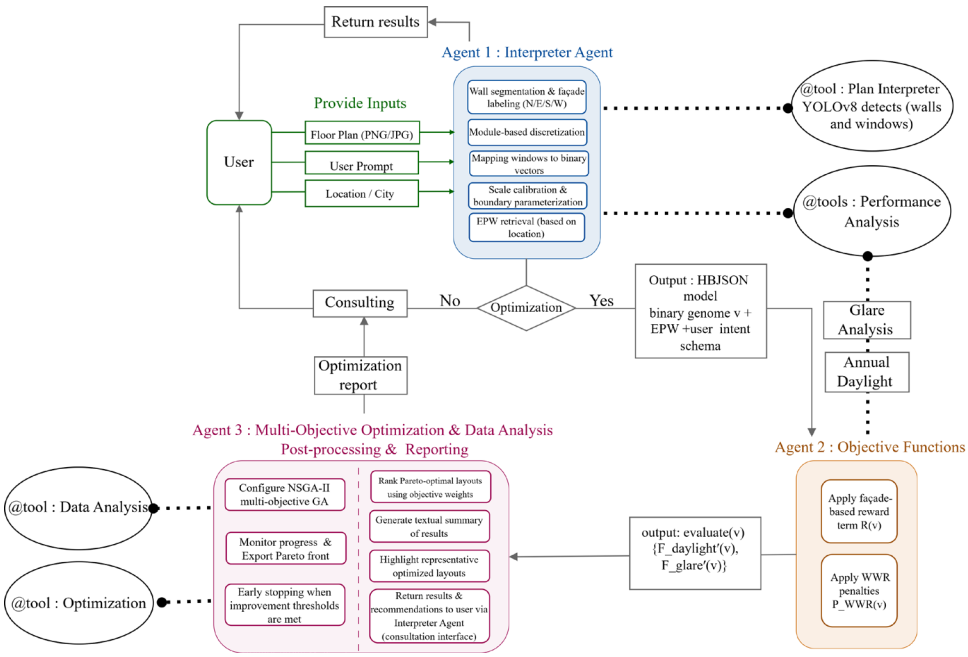


Figure 1.Workflow of the proposed multi-agent LLM framework.

configurations from 2D floor plans, bridging human reasoning and computational intelligence in a way that text-only or parameter-driven optimization approaches cannot. (Figure 1)

2. Methodology

2.1. FRAMEWORK OVERVIEW

This study proposes a three-agent LLM framework that integrates natural-language user intent, computer vision parsing of 2D floor plans, physics-based environmental simulation, and multi-objective optimization, implemented in LangGraph with GPT-4.(LangChain Inc., 2025). Agent A1 captures user intent and converts plans into structured geometry, Agent A2 formulates dynamic objective functions based on performance criteria and user preferences, and Agent A3 performs NSGA-II optimization to generate Pareto-optimal window configurations. The following sections describe the components and present a case study of the full workflow.

2.2. INPUTS

The framework requires a minimal, well-defined set of inputs to ensure robust plan parsing, reproducible simulation, and interpretable optimization. User-provided artifacts include a single-floor architectural raster plan (PNG/JPEG) with clearly defined exterior walls and windows, north alignment, and a known perimeter for scale, along with an EPW weather file that is automatically retrieved based on the user-provided location. Design constraints are specified through global and façade-specific window-to-wall ratio (WWR) bounds (e.g., global WWR $\in$ [0.20, 0.45]; south façade $\in$ [0.15, 0.35]), defining feasible aperture areas, while discrete module or frame sizes (e.g., 0.60 m or 0.90 m with fixed increment steps) reflect fabrication standards (Wright et al., 2014). In addition, qualitative goals are expressed as natural-language statements of comfort or visual objectives, such as maximizing daylight on the south façade or avoiding glare on the west side. These inputs are transformed into a structured schema that guides downstream simulation and optimization.

2.3. A1: INTERPRETER AGENT

The interpreter agent serves as the user-facing interface, capturing design intent and interpreting the floor plan to guide users through analysis or optimization. A GPT-4–based LLM (OpenAI, 2024) conducts multi-turn dialogue to collect inputs, verify scale and orientation, retrieve the EPW file, and elicit façade-specific constraints. When directions are ambiguous (e.g., "exclude the party wall"), the agent requests clarification using cardinal directions (N, E, S, W). Qualitative goals, such as preserving west-facing park views, are then translated into a structured, façade-oriented preference schema, for example:

{"objectives": ["maximize_daylight", "minimize_glare"],
"weights_objectives": {"daylight": 0.6, "glare": 0.4},
"weights_facades": {"N": 0.3, "E": 0.5, "S": 0.2, "W": 0.0},
"wwr_bounds_facades": {"N": [0.15, 0.35], "E": [0.00, 0.2], "S": [0.10, 0.25], "W":

[0.00, 0.00]}}

This schema defines optimization objectives, post-Pareto weighting, façade priorities, hard WWR limits, and discrete module sizes (0.60 m or 0.90 m) for fabrication-based window discretization. Finally, the agent determines whether the user intent corresponds to analysis or optimization mode: in analysis mode, it evaluates the current design and provides natural-language feedback (Section 2.4.2), whereas in optimization mode, it forwards the schema, geometric model, and constraints to downstream agents for multi-objective optimization (Figure 2).

2.4. TOOLS

2.4.1. Plan Interpreter

The Plan Interpreter converts a user-provided raster floor plan into a structured numerical representation of walls and windows for simulation and optimization. Object Detection: A YOLOv8 model(autogyro, 2025) is trained on the CubiCasa5K dataset, detects walls, windows, and doors. Polygon annotations are converted to YOLO-style bounding boxes, retaining only 3 categories. Data are split into training (80%), validation (10%), and test (10%), resized to 1024×1024, and augmented by ±45° rotations(Kalervo Ahtiand Ylioinas, 2019) OCO-pretrained weights, AdamW optimizer, and cosine learning-rate scheduling are used, with performance monitored via COCO-style metrics (mAP@[0.5:0.95], AP_{50})(Lin Tsung-Yiand Maire, 2014). Non-maximum suppression filters detections; wall boxes are reconstructed into polylines, windows snapped to nearest exterior walls, and doors also considered as solid walls. The outer boundary is scaled from the given perimeter and parameterized as a closed curve from a fixed reference corner.

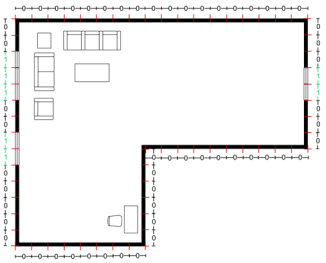
Wall Segmentation & Orientation: Corners split the boundary into wall segments. In this work, we restrict ourselves to orthogonal floor plans. Each segment's outward normal is compared with global axes to assign it to a façade (N, E, S, W).

Module-based discretization: Given a user-defined module size m(e.g., 0.60 m)(Wright et al., 2014), each wall segment i with length $L_i$is discretized into L_i/m one-dimensional slots. For each façade $f \in \{N, E, S, W\}$, this yields a binary vector:

$$\mathbf{v}_f = \left[v_{f,1}, \ldots, v_{f,n_f}\right], v_{f,k} \in \{0,1\},$$

where $v_{f,k} = 0$ denotes opaque wall and $v_{f,k} = 1$ denotes window area at module position $k$on façade f.

Mapping detected windows to modules: Detected windows are mapped onto façade-aligned wall segments as binary vectors (1 = window, 0 = wall), which are concatenated into a single genome used for both optimization and simulation. Window geometry is reconstructed from this encoding to generate Honeybee models and HBJSON files for Ladybug Tools analysis (Figure 2).



Edge	Length (m)	Façade	Binary Vector (window/wall)
1	10.8	N	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	8.4	W	[0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0]
3	4.8	S	[0, 0, 0, 0, 0, 0, 0]
4	3.6	E	[0, 0, 0, 0, 0]
5	6.0	S	[0, 0, 0, 0, 0, 0, 0, 0, 0]
6	4.8	E	[0, 0, 0, 1, 1, 0, 0, 0]

Figure 2.Visualized sample floor plan with numerical representation

2.4.2. Performance Analysis

Performance analysis tools are implemented as Python functions that evaluate binary façade encodings using geometric and climatic data to produce scalar performance metrics. The toolbox may combine physics-based simulations and surrogate models to assess daylight (UDI), glare (DGP-based), solar exposure, energy demand, and view indices, with tools selected based on user goals and computational budget. In this study, two physics-based simulations are used: annual daylight performance via UDI Mean (percent of hours with useful daylight) and glare via a DGP-based indicator (percent of hours with DGP > 0.40), both computed using Honeybee and Ladybug Tools (Python–Radiance). A1 reconstructs apertures from façade encodings, runs the simulations, and returns performance metrics, while downstream agents perform optimization.

2.5. A2: OBJECTIVE FUNCTION AGENT

The The A2 Agent receives the binary façade encoding, the HBJSON model, and the user preference schema from A1. Governed by a GPT-4 LLM, it selects the relevant analysis tools and formulates objective functions that combine simulation outputs, arrangement-based rewards, and constraint penalties, and exposes an evaluate(candidate) function for the downstream optimization agent. For a given candidate window layout $\mathbf{v}$, the Objective Function Agent reconstructs the window geometry from the binary vector and matches each entry in the specified objectives to the corresponding performance analysis tool, such as an annual daylight function (UDI Mean) for “maximize_daylight” or a DGP-based indicator for “minimize_glare.” It then calls the selected tools to obtain base performance metrics, for example $D(\mathbf{v})$: representing daylight performance (to be maximized) and $G(\mathbf{v})$: representing glare (to be minimized). At this stage, these metrics reflect only the physical performance of the layout and remain independent of user orientation preferences.

2.5.1. Arrangement-based reward from façade weights

User orientation preferences are encoded in "weights_facades" as non-negative weights $f \in \{N, E, S, W\}$. These weights do not directly scale daylight or glare values; instead, they define a reward term based on window distribution. For a candidate $\mathbf{v}$, the agent computes:

the number of window modules on each façade, $n_f(\mathbf{v})$, the total number of window modules, $n_{\text{tot}}(\mathbf{v})$, and the fraction of windows per façade:

$$q_f(\mathbf{v}) = \frac{n_f(\mathbf{v})}{\max\left(1, n_{\text{tot}}(\mathbf{v})\right)}$$

An alignment score $A(\mathbf{v})$is then defined as:

$$A(\mathbf{v}) = \sum_f w_f \, q_f(\mathbf{v}),$$

which lies in $[0,1]$: it is high when window area is concentrated on high-weight façades and low when windows are placed on façades that the user de-emphasizes or effectively excludes (weight 0). This alignment score is converted into a reward term:

$$R(\mathbf{v}) = \gamma\, A(\mathbf{v})$$

where γ is a scaling factor chosen so that $R(\mathbf{v})$is of comparable magnitude to the main metrics. The reward is then added to the objectives:

Daylight objective (to maximize):

$$F_{\text{daylight}}(\mathbf{v}) = D(\mathbf{v}) + R(\mathbf{v})$$

Glare objective (to minimize):

$$F_{\text{glare}}(\mathbf{v}) = G(\mathbf{v}) - R(\mathbf{v})$$

so that solutions aligning with the user's façade priorities are favored, even when glare values themselves are similar. Façade weights affect optimization via an additional reward/penalty term based on window arrangement, rather than directly scaling physical performance metrics.

2.5.2. WWR-based hard constraints and penalties

The schema also specifies hard WWR bounds per façade via "wwr_bounds_facades". For each façade f, the agent computes the realized WWR, $\text{WWR}_f(\mathbf{v})$, from the binary encoding and checks:

$$\text{WWR}_f^{\min} \le \text{WWR}_f(\mathbf{v}) \le \text{WWR}_f^{\max}$$

Violations are aggregated into a constraint penalty:

$$P_{\text{WWR}}(\mathbf{v}) = \sum_f \big(\max\,(0, \text{WWR}_f^{\min} - \text{WWR}_f(\mathbf{v}))^2 + \max\big(0, \text{WWR}_f(\mathbf{v}) - \text{WWR}_f^{\max}\big)^2\big).$$

The penalized objectives become:

$$F'_{\text{daylight}}(\mathbf{v}) = D(\mathbf{v}) + R(\mathbf{v}) + \lambda P_{\text{WWR}}(\mathbf{v})$$
$$F'_{\text{glare}}(\mathbf{v}) = G(\mathbf{v}) - R(\mathbf{v}) + \lambda P_{\text{WWR}}(\mathbf{v})$$

with λ a large positive factor that discourages WWR violations. Façades declared as non-window façades are therefore strictly enforced. These penalized objectives are encapsulated in a Python evaluation function within a dedicated module, which the Optimization Agent imports and calls as evaluate(v) during the NSGA-II search.

2.6. A3: DATA ANALYSIS AND OPTIMIZATION AGENT

The A3 Agent performs MOO and result post-processing, guided by a GPT-4–based LLM. It receives the binary vector of the plan (design genome), the preference schema (objectives, objective weights, façade weights, WWR), and the evaluation function evaluate(v) constructed by the A2.

2.6.1. Optimization Tool: NSGA-II

The optimization agent applies a standard NSGA-II search on a façade-wise binary genome (0 = wall, 1 = window), with the initial population generated from the current layout and random feasible variants under WWR constraints. New candidates are produced via mutation and crossover, and populations are ranked using non-dominated sorting on penalized daylight and glare objectives, with diversity preserved through crowding distance.

2.6.2. Data analysis

During optimization, the Data Analysis Tool summarizes the evolving design space. It tracks convergence indicators such as hypervolume gains and façade-layout diversity, and groups Pareto-optimal designs into a small number of representative families (e.g., balanced, daylight-oriented, glare-averse). t returns compact tables and ranked shortlists, which A3 packages into short textual reports for A1. A1 then uses these reports in the dialogue with the user (e.g., highlighting that low-glare options darken the core or that most windows cluster on one façade).

2.6.3. Online monitoring and early stopping

For online monitoring, A3 calls the tool every fixed number of generations (e.g., 10) and exports the current Pareto set as a table with façade-level WWRs, daylight and glare metrics, and reward/penalty terms, evaluating improvement and diversity. Instead of automatic stopping, A3 passes the summary to A1, which informs the user and asks whether to continue, stop, or adjust constraints (e.g., shifting emphasis between daylight and glare or excluding façades). A1 updates objective weights, bounds, or filters, while A2/A3 adjust the evaluation function and search; if trade-offs are accepted and further progress is negligible, A3 triggers early stopping and freezes the final Pareto set.

3. Result

3.1. YOLO-BASED DETECTION MODEL PERFORMANCE

The YOLO-based detector achieved strong performance on the validation set, with 74.0% precision, 81.2% recall, and mAP scores of 80.5% (mAP@0.5) and 52.5% (mAP@0.5:0.95). Training curves over 120 epochs show steadily decreasing losses and stabilizing mAP, indicating good convergence without notable overfitting. The normalized confusion matrix further confirms high class-wise accuracy (up to 96% true positives for some classes). Overall, these results demonstrate that the trained YOLO model is efficient and reliable for the target detection task.

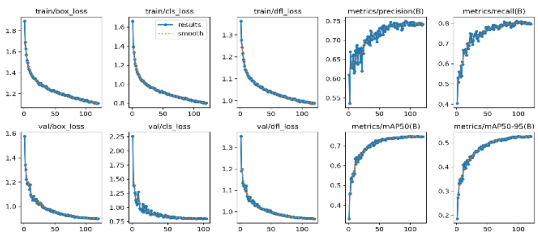


Figure 3 .training process

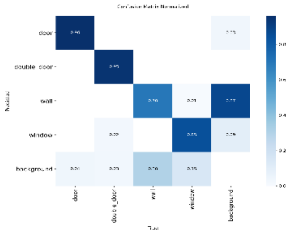


Figure 4 .confusion matrix

3.2. OPTIMIZATION WORKFLOW AND PARETO FRONT GENERATION

To validate the framework, a representative case study was conducted. Agent A1

translated a 2D floor plan and the user query "I need strong daylight for the main area, but I must avoid glare on the west wall where the computer screens are" into a weighted preference schema prioritizing daylight (wd = 0.7) with a façade-specific penalty on west-facing glare. Agent A2 instantiated the evaluation function, and Agent A3 executed NSGA-II while tracking the evolving Pareto front (Figure 5). Unlike conventional workflows that only return a final set of non-dominated solutions, Agent A3 continuously analyzes convergence, diversity, and schema compliance and reports concise summaries to Agent A1, enabling conversational steering. For example, "A3 reports that most low-glare layouts significantly darken the core. Would you accept slightly higher glare on the west wall to brighten the main area?" The user may respond, "Yes, a bit more glare is acceptable if the main area feels brighter overall." Agent A1 then updates the preference schema, prompting Agents A2 and A3 to adjust objectives or trigger early stopping once satisfactory solutions are reached. In the final stage, Agent A3 clusters the Pareto front and recommends a small set of representative designs. Users may further refine results through dialogue (e.g., "Now only show options without windows on the north façade and with compact window patterns"), which is applied as filters to the existing Pareto set, enabling adaptive decision-making rather than static post-processing. To evaluate practical utility, a comparative user study with ten Master's students in Building Energy Engineering compared the proposed framework against a manual MOO workflow and a fixed baseline (UDI = 55%, DGP = 45%) using the same 2D plan and objectives (Figure 6). While the conventional workflow required approximately 450 minutes of computation plus 30 minutes of manual filtering, the proposed framework enabled acceptable solutions in an average of 180 minutes.

Moreover, 40% of participants triggered early stopping and reached satisfactory solutions within 45–115 minutes. Although these solutions occasionally showed slightly lower performance than full optimization, the average time reduction of about 80% (up to 90% for some users) represented a favorable trade-off. Continuous exposure to intermediate Pareto fronts (updated every ~10 seconds) further supported real-time exploration and revealed alternatives unlikely to emerge from static Pareto inspection alone.

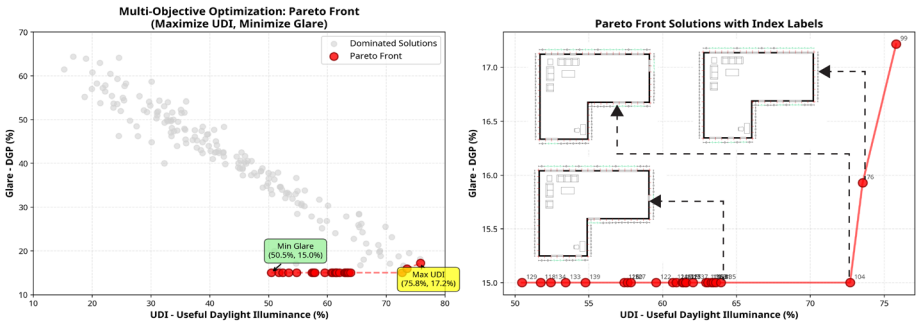


Figure 5 .Pareto front after 50 iterations with three optimized architectural layouts.

4. Discussion, Limitations, and Future Works

The results indicate that the proposed framework substantially outperformed the conventional manual MOO workflow in terms of time efficiency while maintaining competitive performance outcomes. The success of participants U1–U6, who completed the full optimization cycle and often surpassed manual MOO performance, confirms that the preference schema generated by Agent A1 effectively translates qualitative design intent into quantitative optimization weights (wd, wg). The conversational interaction further enabled users to explore solution spaces that are typically inaccessible in static optimization setups. For approximately 20% of participants, dialogue-driven steering redirected the search toward alternatives absent from the original static Pareto set, demonstrating that the framework actively engages users in the optimization process rather than merely post-filtering results.

The current prototype is limited to 2D floor plans, daylight and glare performance metrics, and window layouts discretized into fixed module sizes, which restricts direct applicability to multi-storey, volumetric, or highly irregular façades. Additional limitations include the reliability of LLM reasoning and the potential for error propagation across interacting agents. Future work will extend the framework to 3D geometry and volumetric simulations, integrate additional performance criteria such as thermal comfort and energy consumption, and explore adaptive learning mechanisms that refine preference schemas based on repeated user interactions over time.

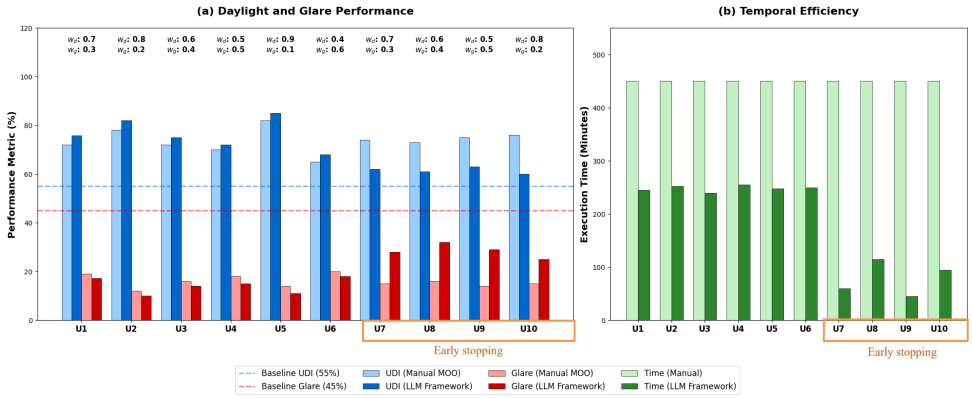


Figure 6 .Quantitative comparison of manual MOO vs. proposed LLM framework.

Attribution Statement

ChatGPT (OpenAI, 2024) was used to refine grammar, wording, and flow.

References

autogyro. (2025). yolo-V8. GitHub. https://github.com/autogyro/yolo-V8

Bagasi, O., Nawari, N. O., & Alsaffar, A. (2025). BIM and AI in Early Design Stage: Advancing Architect–Client Communication. Buildings, 15(12).

Chi, D. (2021). An Approach to Determine Specific Targets of Daylighting Metrics and Solar Gains for Different Climatic Regions. Journal of Daylighting, 8, 1–19.

Diakaki, C., Grigoroudis, E., & Kolokotsa, D. (2008). Towards a multi-objective optimization approach for improving energy efficiency in buildings. Energy and Buildings, 40(9), 1747–1754.

Hou, J., Qiu, K., Zhang, Z., Yu, Y., Wang, K., Capolongo, S., Zhang, J., Li, Z., & Zhang, J. (2024). Wireless-Friendly Window Position Optimization for RIS-Aided Outdoor-to-Indoor Networks based on Multi-Modal Large Language Model. CoRR, abs/2410.20691.

Jiang, G., Ma, Z., Zhang, L., & Chen, J. (2024). EPlus-LLM: A large language model-based computing platform for automated building energy modeling. Applied Energy, 367, 123431.

Kalervo Ahti and Ylioinas, J. and H. M. and K. A. and K. J. (2019). CubiCasa5K: A Dataset and an Improved Multi-task Model for Floorplan Image Analysis. In P.-E. and S. I.-M. and U. J. Felsberg Michael and Forssén (Ed.), Image Analysis (pp. 28–40). Springer International Publishing.

Konstantzos, I., Tzempelikos, A., & Chan, Y.-C. (2015). Experimental and simulation analysis of daylight glare probability in offices with dynamic window shades. Building and Environment, 87, 244–254.

LangChain Inc. (2025). LangGraph. https://github.com/langchain-ai/langgraph

Lin Tsung-Yi and Maire, M. and B. S. and H. J. and P. P. and R. D. and D. P. and Z. C. L. (2014). Microsoft COCO: Common Objects in Context. In T. and S. B. and T. T. Fleet David and Pajdla (Ed.), Computer Vision – ECCV 2014 (pp. 740–755). Springer International Publishing.

Liu, H., Zhang, Z., Ma, X., Lu, W., Li, D., & Kojima, S. (2021). Optimization Analysis of the Residential Window-to-Wall Ratio Based on Numerical Calculation of Energy Consumption in the Hot-Summer and Cold-Winter Zone of China. Sustainability, 13(11).

Postle, B., & Salingaros, N. A. (2025). LLM and Pattern Language Synthesis: A Hybrid Tool for Human-Centered Architectural Design. Buildings, 15(14).

Trząski, A., & Rucińska, J. (2025). Integration of Daylight in Building Design as a Way to Improve the Energy Efficiency of Buildings. Energies, 18(15).

Wright, J. A., Brownlee, A., Mourshed, M. M., & Wang, M. (2014). Multi-objective optimization of cellular fenestration by an evolutionary algorithm. Journal of Building Performance Simulation, 7(1), 33–51.

Zhang, L., & Chen, Z. (2025). Opportunities of applying Large Language Models in building energy sector. Renewable and Sustainable Energy Reviews, 214, 115558.

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

Zhao, J., & Du, Y. (2020). Multi-objective optimization design for windows and shading configuration considering energy consumption and thermal comfort: A case study for office building in different climatic regions of China. Solar Energy, 206, 997–1017.