2026 European Conference on Computing in Construction
Corfu, Greece
July 12–15, 2026

EXPLORING ADAPTABLE OFFICE FIT-OUT LAYOUTS USING CONFIGURABLE FLOOR PLAN GRAPHS AND LARGE LANGUAGE MODELS

Mikhael Johanes[1], Simona Kocour[1,2], Siyu Tang[1], and Catherine De Wolf[1]
[1]ETH Zurich, Zurich, Switzerland
[2]CTU in Prague, Czech Republic

Abstract

Conventional office fit-outs are designed for fixed requirements, which limits their long-term adaptability. This paper introduces a workflow that integrates configurable floor-plan graphs with large language models (LLMs) to enable adaptable office layout exploration. Interior space is represented as a graph that encodes geometry, adjacency, and accessibility, and is parameterised by binary partition decisions. For each candidate configuration, an LLM infers program assignments across multiple scenarios, and solutions are evaluated in a multi-objective combinatorial optimisation. Results show configurations with feasible performance across scenarios, suggesting that LLM-based program assignment can serve as a flexible semantic component that complements quantitative solver-based methods.

Introduction

Offices are constantly evolving. Changes in organisational structures, workforce patterns, and work styles require interiors to adapt. However, this adaptation often comes with a cost: frequent renovations lead to significant environmental impacts, with emissions and waste from repeated fit-out cycles exceeding those of the building itself during its lifetime (Casas Arredondo, 2021; Forsythe and Wilkinson, 2015).

Circular fit-out design has emerged as a response to the environmental consequences of short fit-out lifecycles. In circular construction, adaptability is a core design requirement that increases the ability to accommodate changes in use, environment, and specification throughout time without extensive refurbishment, enabling longer building lifespans and lower embodied greenhouse gas emissions (Watt et al., 2023). Rather than optimising interiors for a single functional state, adaptable fit-outs explicitly account for change over time, by extending the design scope to encompass successive use scenarios within the same spatial system.

Computational generative-design methods offer a promising means to support such adaptability by systematically generating and evaluating alternative spatial configurations under varying requirements. However, most existing generative-design paradigms remain orientated toward single-scenario or fixed specifications. Implicitly assuming stable functional requirements, they optimise layouts for a single, fixed programmatic state. As a result, adaptability is typically treated as an evaluative afterthought rather than a generative principle embedded in the design process itself.

This research addresses the limitation of single-scenario design by decoupling spatial configuration from programmatic assignment. We investigate how floor plan graphs, in which spaces are modelled as nodes and connectivity is encoded as edges, can support large language models (LLMs) in assigning program labels for a given spatial configuration under different scenarios. Multi-objective optimisation is employed to find configurations that maintain acceptable performance across multiple programmatic scenarios.

The hypothesis is threefold. First, separating spatial configuration from programmatic assignment enables the modelling of multifunctionality, in which a single space can accommodate multiple uses. Second, floor plan graphs act as an intermediate representation bridging geometric and semantic interpretation. Third, LLM-based programmatic assignment may provide a flexible semantic labelling step that complements quantitative constraint-based solvers and helps relate underspecified preferences to optimisation objectives.

The proposed workflow supports finding spatial configurations that can accommodate multiple programmatic scenarios (Figure 1). By operationalising adaptability in the workflow, this work provides a method for exploring spatial configurations that remain viable across multiple future use scenarios, supporting longer fit-out lifecycles.

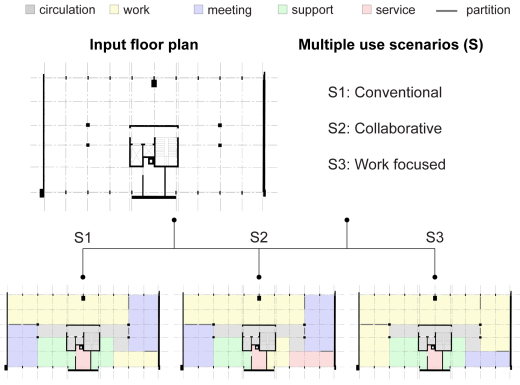

Figure 1: Adaptive layout objective. Given an input floor plan and multiple scenario requirements, we seek layouts that accommodate multiple programs without reconfiguration.

Background

Modelling spatial adaptability in circular fit-out

Adaptability operates on multiple temporal scales, ranging from short-term adjustments such as moving furniture, through medium-term reconfiguration of spatial layouts, to long-term transformations including changes in building use (Schmidt-III and Austin, 2016; Watt et al., 2023). For office interiors, adaptability is primarily situated on a short- to medium-term scale, where frequent organisational and operational changes require interior interventions. To minimise environmental impact, we posit that adaptability at these scales depends on the latent capacity of a spatial layout to support multiple configurations with minimal structural intervention.

The spatial capacity represented by the dimensions of space and the position of durable elements plays a significant role in the definition of the future adaptability (Durmišević, 2018). Graph-based representations offer an opportunity to model such spatial capacity. For example, Pacquée et al. (2025) developed a graph-based method to encode the topology and geometry of a building, allowing comparison between reference projects to assess their potential for future reuse. At the scale of architectural layouts, Herthogs et al. (2019) modelled floor plans as spatial graphs and quantified their capacity for change through measures of topological depth and path redundancy as a proxy of the layout's ability to support multiple programmatic scenarios over time.

Building on this body of work, we adopt spatial graphs not primarily as an evaluative tool to measure adaptability but as a configurable representation to systematically explore alternative layout configurations under varying programmatic scenarios.

Generative design of an adaptable office

Generative spatial layout design has been widely studied in computational design and space-planning research, which commonly distinguishes between knowledge-driven and data-driven approaches (Yan et al., 2025). Knowledge-driven methods generate layouts by explicitly structuring the design through rules, constraints, and objectives. They encompass rule-based systems, heuristic and mathematical optimisation, and reinforcement learning. Architectural intent is explicitly encoded at different stages of the generative process: rule-based methods formalise the design intent through predefined spatial rules and discretization schemes (Sheijani et al., 2024), parametric workflows couple parametric models with multi-objective evolutionary optimisation to explore trade-offs among evaluation criteria (Nagy et al., 2017), and hybrid approaches combine constraint programming with evolutionary search to generate feasible layouts within fixed envelopes (Laignel et al., 2021). More recently, reinforcement learning has been used to incorporate spatial constraints and objectives directly into state, action, and reward formulations (Zhang et al., 2024b). Although considerably effective, these approaches rely on extensive explicit knowledge encoding, and thus face scalability limits for real-world problems that are often open-ended and incomplete.

Data-driven approaches generate spatial layouts by learning patterns from given datasets; for example, they map partial design input such as text prompts, bubble diagrams, or footprint constraints to complete spatial layouts. Data-driven approach includes statistical generative models, deep neural networks, and example-based or case-based generation methods. Representative examples include models based on the generative adversarial network (GAN) such as HouseGAN (Nauata et al., 2020), diffusion-based approaches such as HouseDiffusion (Shabani et al., 2023), and transformer-based models such as MaskPLAN (Zhang et al., 2024a). However, the limited availability of large, high-quality architectural datasets has mostly confined these methods to residential typologies.

Beyond these limitations, most generative-design methods optimise layouts for a single design state, limiting future adaptability. We address this limitation by decoupling spatial configuration from programmatic assignment. This separation allows multiple program interpretations to be inferred from the same spatial configuration in response to different use scenarios.

LLM graph-based spatial reasoning in generative design

Recent advances in LLMs have shown growing capabilities in symbolic and relational tasks, motivating their integration into generative-design workflows for spatial layout problems. Rather than directly generating precise geometry, most approaches use LLMs as high-level semantic or planning components operating over abstract spatial representations, such as graphs, symbolic relations, or procedural descriptions. At the interior scale, LayoutGPT frames layout generation as a compositional planning problem using in-context examples, in which LLMs translate natural language prompts into structured descriptions of objects and their spatial properties in a CSS-like format (Feng et al., 2023). FlairGPT uses LLMs to generate zones, objects, and inter-object constraints, which are organised into a layout constraint graph and solved using standard optimisation algorithms (Littlefair et al., 2025). At the building scale, Text2BIM extends this direction through a multi-agent framework that generates structured procedural instructions for BIM authoring tools (Du et al., 2026). Collectively, these works suggest that LLMs are most effective when applied to high-level semantic and relational tasks, while detailed geometric realisation is delegated to rule-based, procedural, or optimisation-based systems.

Method for graph-based fit-out design explorations

Overview

We propose a generative workflow for an adaptable layout design that integrates spatial-configuration optimisation with scenario-driven programmatic assignment (Figure 2). The workflow starts with two inputs: an input floor

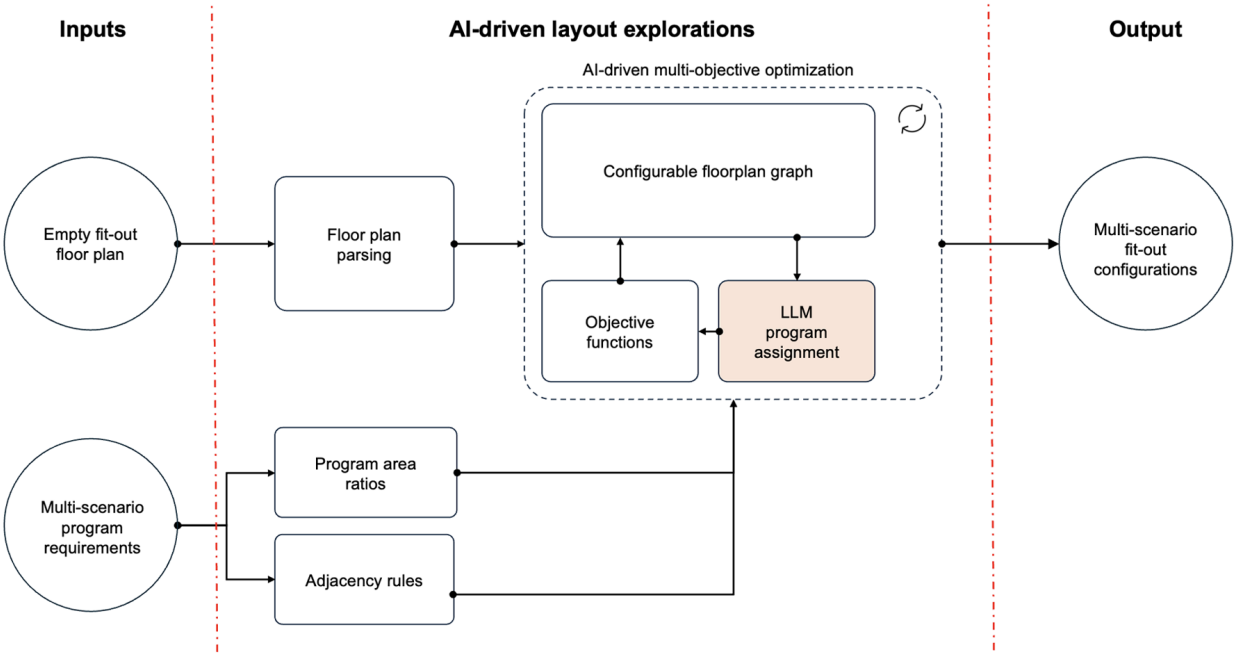

Figure 2: Overview of graph-based fit-out layout explorations with LLM

plan and a set of use scenarios. The floor plan is parsed into a configurable floor plan graph that defines a fixed topological layout independent of the program. The programmatic requirements of each scenario are delegated to an LLM, which assigns zone labels to the floor plan graph according to the area-ratio targets and adjacency rules. A multi-objective evolutionary algorithm optimises the spatial configuration using these criteria, yielding layouts that can accommodate multiple scenarios.

Parsing the input floor plan into a configurable graph

The input floor plan contains geometric information on the existing structure, walls, doors, and windows as well as predefined zones, such as the entrance and the service areas, and an underlying planning grid used to discretise the layout. The parsing module converts this input into a parametric representation by discretising the floor area into cells and introducing binary decision variables, x, that govern the placement of partitions between adjacent cells (Figure 3). From this representation, an intermediate visibility graph is constructed, and strongly connected cells are clustered to produce a configurable floor plan graph, $G_{fp}(x)$. Changing x, the optimisation algorithm explores alternative spatial configurations, each of which is reflected in a corresponding instance of $G_{fp}(x)$. The resulting graph representation then serves as input for the LLM modules to infer programmatic assignments under different use scenarios.

Programmatic requirements for multi-scenario inputs

Programmatic requirements are specified as a set of discrete use scenarios, each of which is translated into two complementary specifications: (1) target program area ratios and (2) adjacency rules between functional zones.

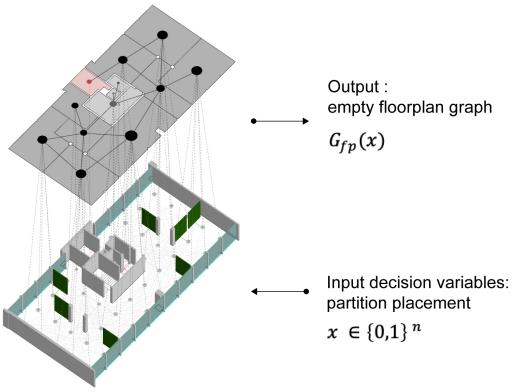

Figure 3: The empty floor plan is parsed into a configurable graph, where binary decision variables defined on the planning grid control the placement of partitions.

This formulation allows multiple future programs to be evaluated against a shared spatial configuration. In this preliminary study, each scenario defines target area ratios for five abstract zone types: Work (W), Meeting (M), Support (Su), Service (Se), and Circulation (C) (Table 1). In parallel, qualitative spatial relationships are encoded as hop-based constraints on the floor plan graph (Table 2). Required adjacencies (R_1) and forbidden adjacencies (F_1) are enforced as hard constraints, defined in terms of graph distance between zones, while preferred proximities (P_2) are modelled as soft objectives that bias solutions toward functional coherence without over-constraining the layout. Circulation is constrained to maintain direct adjacency to all other zone types, ensuring global accessibility across configurations.

For each scenario, s, the programmatic requirements are

represented by a pair (A_s, R_s), where A_s specifies the ratios of the target program area, and R_s specifies the adjacency rules. These scenario specifications serve two functions in the workflow. First, they are translated into a structured prompt context for the LLM, which assigns zone labels to the nodes of the input floor plan graph for each scenario. Second, they parameterise the objective functions used during evolutionary optimisation. Specifically, deviations from A_s and violations of R_s are quantified through scenario-wise evaluations, which are subsequently aggregated across scenarios to assess the adaptability of a given spatial configuration.

Table 1: Program scenarios as target area ratios (sum = 1.0)

Scenario	W	M	Su	Se	C
S1	0.50	0.20	0.10	0.10	0.10
S2	0.40	0.30	0.10	0.10	0.10
S3	0.65	0.10	0.10	0.05	0.10

Table 2: Adjacency and proximity constraints encoded as hop-based relations

From\To	W	M	Su	Se	C
W	–	P_2	P_2	F_1	R_1
M	P_2	–	–	F_1	R_1
Su	–	–	–	P_2	R_1
Se	F_1	F_1	P_2	–	R_1
C	R_1	R_1	R_1	R_1	–

LLM programmatic inference

Given a candidate floor plan graph, G_{fp}, and a scenario-specific program configuration (A_s, R_s), we construct a textual prompt, P_s, consisting of a program block and a rules block. The floor plan graph, G_{fp}, is provided to the LLM as a structured JSON representation encoding global, node, and edge properties, including spatial attributes such as area, perimeter components, bounding boxes, aspect ratios, and interface characteristics between adjacent spaces. The program block derives the total usable area from the node attributes in G_{fp} and specifies the target area for each zone type in A_s, reported in absolute and relative terms. The rules block serialises the adjacency constraints in R_s as hard and soft relations with associated predicates and targets. A structured system instruction and output template guide the LLM to assign zone labels within a predefined schema without modifying the underlying spatial topology. The instruction embeds the scenario context, P_s, together with the list of valid zone types. Outputs that do not conform to the expected schema or label set are discarded, and the original unlabelled floor plan graph is retained for evaluation. Analysis of the reported experiments suggests that such fallback was negligible, occurring in at most 1 of 5000 evaluated rows for the LLM-based setups. The final prompt is formed by concatenating the two blocks (1), and the LLM receives G_{fp} and P_s to assign zone labels, producing an updated graph, G_{fp}^{llm} (2).

$$P_s(G_{fp}, A_s, R_s) = P_{\text{prog}}(G_{fp}, A_s) \parallel P_{\text{rules}}(R_s) \quad (1)$$

$$G_{fp}^{\text{llm}} \leftarrow \text{LLM}(G_{fp}, P_s) \quad (2)$$

Multi-objective optimisation

Optimisation variables are the binary partition decisions, x, defined in the planning grid. For each candidate, x, a floor plan graph, $G_{fp}(x)$, is generated and tested for connectivity. Only connected configurations invoke LLM inference; disconnected graphs are excluded using a hard feasibility constraint.

For each scenario, $s \in S$, the LLM assigns zone labels to $G_{fp}(x)$ using the corresponding program area ratios and adjacency rules, producing a set of inferred graphs, $\{G_{fp}^{\text{llm}}(s)\}$. Then, two objective functions are evaluated and aggregated across scenarios: program-ratio fitness and rule compliance. The fitness of the program ratio measures the deviation between the achieved and target area ratios using the mean absolute error (MAE). Rule compliance evaluates the adherence to required, forbidden, and preferred adjacencies using hop-based scoring, normalised to a score in the range $[0, 100]$. The resulting bi-objective optimisation problem is defined as:

$$f_1(x) = \frac{1}{|S|} \sum_{s \in S} \text{MAE}_{\text{area}}\left(G_{fp}^{\text{llm}}(s), A_s\right), \quad (3)$$

$$f_2(x) = -\frac{1}{|S|} \sum_{s \in S} \text{Score}_{\text{rules}}\left(G_{fp}^{\text{llm}}(s)\right), \quad (4)$$

$$\text{subject to } G_{fp}(x) \in \mathcal{G}_{\text{conn}}. \quad (5)$$

We solve this bi-objective optimisation problem using NSGA-II (Deb et al., 2002). The initial population is sampled as sparse binary vectors to promote diversity in partition configurations. Optimisation terminates after a fixed number of generations or when changes in decision variables and objective values fall below a predefined tolerance.

Case study and discussion

Based on an input office floor plan with a total area of approximately 400 m^2 (Figure 1) and the multi-scenario programmatic specifications in Tables 1 and 2, we conducted a series of experiments to evaluate the proposed workflow using different LLMs and a baseline approach, assessing their performance across optimisation quality, spatial regularity and topological complexity of the resulting layouts.

Experiment setup

LLM-based programmatic inference is executed using a locally hosted Ollama server accessed through the Python `ollama` client, keeping all inference local to the experimental environment. We evaluate three large language

models with increasing capacity: `phi4:14b`, `deepseek-r1:32b`, and `gpt-oss:120b`, where model size (in billions of parameters) provides a rough indication of overall model capability.

To ensure reproducibility, deterministic decoding is used where possible. Greedy decoding (temperature = 0.0) is applied for `phi4:14b` and `deepseek-r1:32b`, producing consistent outputs across runs. For `gpt-oss:120b`, temperature = 0.0 proved unstable in our Ollama setup; therefore, we retain the model's default sampling configuration (temperature = 1.0) and set `thinking=low`, leaving other decoding parameters unchanged.

As a baseline, we implement a constraint-programming model using OR-Tools CP-SAT (Perron et al., 2023). Following established discrete space allocation formulations (Laignel et al., 2021), the solver assigns a single zone label to each node of the floor plan graph using a one-hot encoding while keeping the topology fixed and respecting predefined labels. The objective minimises deviations from target area ratios and penalties for soft spatial preferences, such as proximity and path coherence. Hop-based adjacency rules are converted into weighted penalties. Circulation accessibility is approximated by enforcing connectivity among circulation-labelled nodes using a predecessor-tree formulation.

Optimisation of spatial configuration is performed using NSGA-II implemented in an existing Python package (Blank and Deb, 2020), with a population size of 50 evolved over 100 generations. The initial population is generated through Bernoulli sampling with an open-edge probability of 0.1, biasing the search toward sparse partition configurations in early generations. Results are reported from a single optimisation run per setup with a fixed random seed. For each optimisation run, LLM inference consumed an estimated 49.9 million tokens in total, including 46.0 million input tokens and 3.9 million output tokens.

We evaluate the results in three complementary dimensions. First, optimisation performance is assessed by comparing Pareto fronts across setups using hypervolume (HV) (Fonseca et al., 2006) and inverted generational distance plus (IGD+) (Ishibuchi et al., 2015). Both indicators are computed on normalised objective values, with objectives scaled to $[0, 1]$ across all experiments. For HV, a fixed reference point of $(1.1, 1.1)$ is used to ensure dominance over all feasible solutions. IGD+ is computed with respect to the union of non-dominated solutions obtained across all setups, serving as a common reference set for convergence comparison. These metrics jointly quantify trade-off quality and convergence with respect to program area-ratio fitness and rule compliance. Second, the shape complexity of the generated layouts is evaluated following Brinkhoff et al. (1995). The measure captures the irregularity of the boundaries and the convexity of the shapes of individual rooms, weighted by room area, providing an indicator of spatial regularity and constructibility. Third, the topological complexity of Pareto-optimal floor plan graphs is evaluated using the cyclomatic number (McCabe, 1976), which captures loop redundancy in adjacency and circulation networks and indicates the potential for alternative routing and future adaptability.

Pareto fronts

The comparison of Pareto fronts across different setups is shown in Figure 4 and Table 3. A clear distinction emerges between constraint-based and LLM-based programmatic inference. The CP-SAT baseline consistently achieves higher HV and lower IGD+, reflecting its ability to exploit a fully specified optimisation model under explicitly encoded objectives. In contrast, all LLM-based setups exhibit lower Pareto-front quality, with broader but less extreme fronts, indicating weaker convergence within the defined objective space.

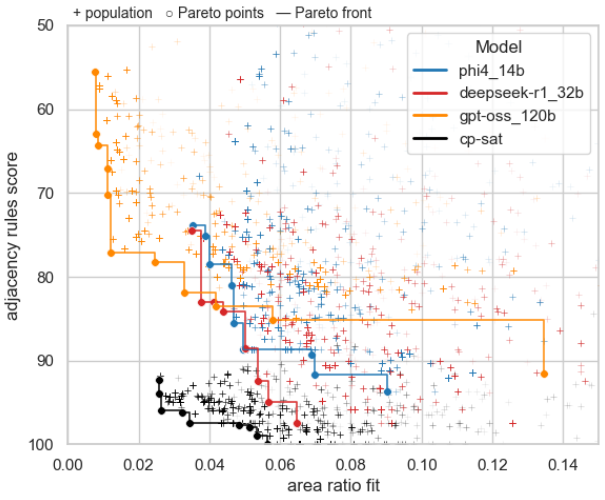

Figure 4: Pareto fronts from the optimisation results

Table 3: Pareto-front quality metrics across setups

Setup	HV ↑	IGD+ ↓
Phi-4	1.051	0.079
DeepSeek-R1	1.092	0.060
GPT-OSS	1.086	0.070
CP-SAT	1.153	0.018

Among the LLM-based configurations, optimisation performance does not consistently improve with larger model size. DeepSeek-R1:32B achieves the highest HV and lowest IGD+ among the LLM variants, suggesting a more balanced trade-off between program area-ratio fitness and adjacency rule compliance. GPT-OSS:120B tends to favour program area-ratio fitness, achieving lower area deviation at the expense of adjacency rule compliance, while Phi-4:14B shows the weakest overall performance across both objectives. Overall, LLM-based programmatic inference remains substantially below the CP-SAT baseline and is better understood as a flexible program-assignment mechanism rather than a replacement for exact optimisation.

These observations point to a hybrid direction for future work, in which hard constraints are handled by classical

solvers while LLMs support more qualitative or underspecified aspects of program assignment.

Floor plan complexity

To further characterise the generated layouts, we use shape complexity and cyclomatic number as proxies for geometric and topological complexity, respectively. Both metrics are evaluated for the Pareto-optimal solutions of each setup, with their distributions shown in Figure 5.

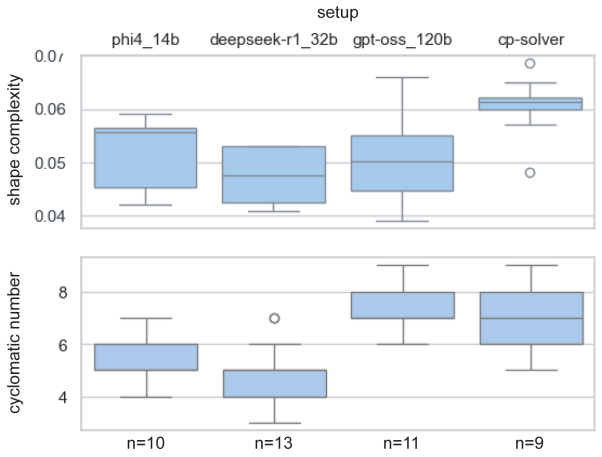

Figure 5: Comparison of room shape and topological complexity of Pareto-optimal floor plan graphs generated by different setups

The results reveal clear differences between the CP-SAT baseline and the LLM-based setups. While the CP-SAT solver identifies Pareto-optimal solutions that more closely satisfy the explicitly defined objectives, it also produces layouts with more irregular room geometries. This indicates a geometric trade-off associated with closely fitting the optimisation objectives.

A separate pattern appears in the topological metrics. The higher distribution of cyclomatic complexity indicates that CP-SAT also explores configurations with more complex adjacency and circulation structures. This is broadly consistent with (Herthogs et al., 2019), who related spatial-graph measures such as topological depth and path redundancy to a layout's capacity for change across programmatic scenarios. From this perspective, higher topological complexity may indicate greater capacity for change across programmatic scenarios, as increased loop redundancy can support alternative spatial relationships.

In contrast, LLM-based setups tend to favour structurally simpler layouts. Phi-4:14B and DeepSeek-R1:32B both produce layouts with simpler room geometries and lower cyclomatic complexity, with DeepSeek-R1:32B showing the strongest reduction in complexity. However, from the perspective of graph-based adaptability, this simplification may also imply lower path redundancy and reduced capacity to accommodate alternative programmatic scenarios. GPT-OSS:120B departs from this pattern, exhibiting higher topological complexity and greater variability in room shape complexity, suggesting that stronger optimisation of program area ratios may again introduce more complex spatial structures.

Multi-scenario layout configurations

Figure 6 presents representative multi-scenario layout configurations obtained from each setup. Each column shows a single spatial configuration evaluated in three scenarios, together with summary metrics on geometric complexity, topological complexity, and objective performance. The qualitative observations are consistent with the quantitative analysis reported earlier.

The CP-SAT baseline is a strong benchmark when program requirements are clearly defined. By directly optimising area ratios and adjacency constraints, it consistently achieves the best objective performance, underscoring the value of explicit formulations: when objectives and constraints can be clearly encoded, exact optimisation is most effective. In contrast, LLM-based setups use floor plan graph features and in-context prompting to assign program labels across scenarios. They do not reliably match CP-SAT's objective values and show variable geometric and topological complexity, but they offer an alternative assignment method when scenario-specific formulations are underspecified or difficult to encode explicitly.

Representative samples from `phi4:14b` and `deepseek-r1:32b` balance objective performance with relatively low topological complexity, with `phi4:14b` showing higher geometric complexity than `deepseek-r1:32b`. `gpt-oss:120b` instead produces layouts with lower geometric complexity but more rooms and connections, increasing topological complexity.

Separating spatial configuration search from programmatic assignment allows adaptable layout exploration, since one configuration can be evaluated across scenarios while optimisation targets layouts that support high-quality assignments. In this setup, explicit constraint solving is effective when program requirements are clearly specified, as in the CP-SAT baseline. LLM-based inference is generally less optimal but provides an alternative when scenario requirements are less fully specified or more difficult to encode explicitly.

Limitations

The proposed workflow is intended as an exploratory prototype and several limitations should be addressed in future work. Although the graph-based representation enables LLM-based programmatic inference, a clear performance gap remains compared to the constraint-based baseline, and embedding LLM inference within the optimisation loop introduces substantial computational overhead. This finding motivates more selective hybrid workflows in which well-defined objectives and hard constraints are handled by solvers, while LLM-based programmatic inference is applied only to a smaller set of promising candidate configurations. In addition, the reported experiments are limited to a single optimisation run on one small office floor plate and a set of simplified programmatic sce-

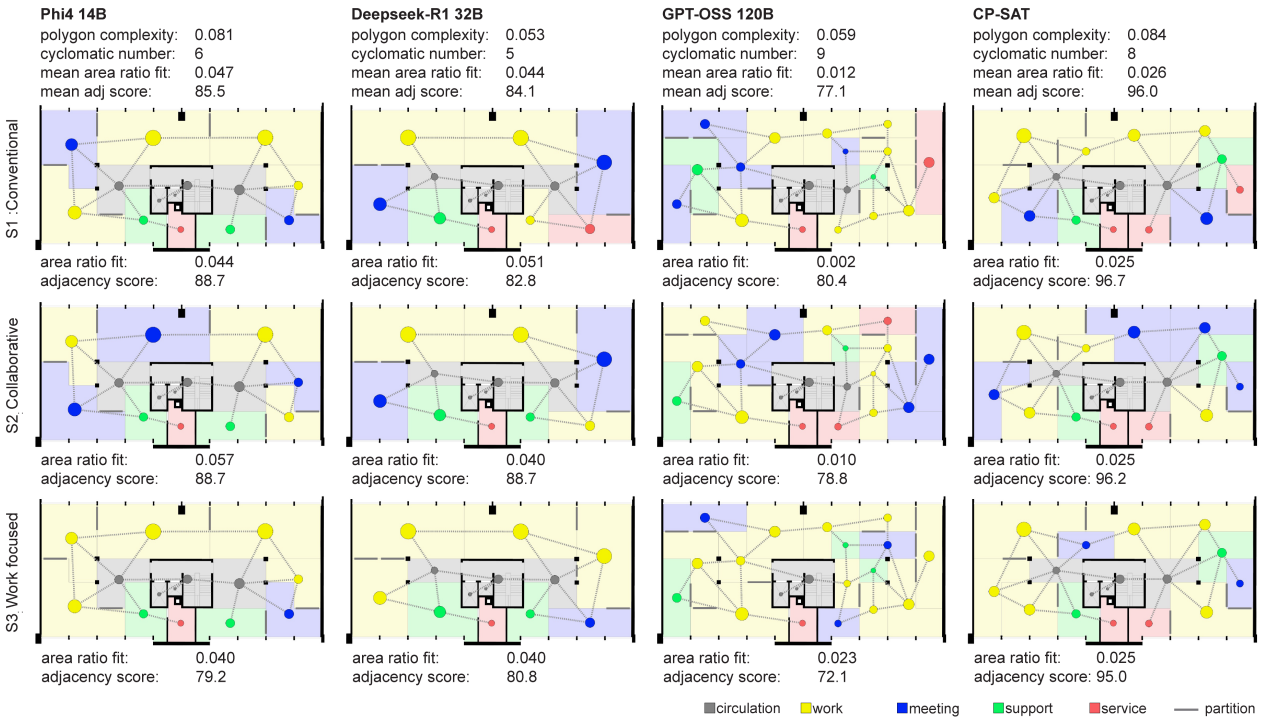

Figure 6: Representative sample of generated multi-scenario floor plans from across setups

narios with only modest structural differences, so the findings should be read as exploratory rather than generalisable. Finally, the evaluation focuses on proxy metrics to assess the feasibility of the results; more comprehensive and practice-orientated measures, including direct indicators of adaptability, circularity, and environmental impact, are required to evaluate the effectiveness of the method in reducing the embodied greenhouse gas emissions associated with office refurbishments.

Conclusion and outlook

This paper presented a graph-based computational workflow for exploring adaptable office fit-out layouts across multiple future use scenarios. By decoupling spatial configuration from programmatic assignment and delegating scenario-specific program assignment to LLMs, the approach supports the systematic exploration of spatial layouts that remain viable across changing requirements. The results show that, while LLM-based programmatic inference does not match the optimisation performance of fully specified constraint-based models, it can generate diverse configurations without requiring exhaustive re-encoding of programmatic constraints for each scenario. This suggests that LLMs are better understood as flexible semantic components within a hybrid optimisation workflow rather than as replacements for exact optimisation.

Looking ahead, the findings point toward more selective hybrid generative-design workflows in which classical optimisation methods enforce hard constraints and quantitative objectives, while LLM-based programmatic inference is applied to a reduced set of promising candidate layouts. Future work should extend the method to larger and more heterogeneous floor plans, richer and more realistic program definitions, and a broader set of evaluation criteria relevant to adaptability, circularity, and environmental impact. Incorporating downstream fit-out feasibility and reuse-related assessments would further support practical application in circular fit-out design.

Acknowledgements

This work was supported through a Design++ fellowship program at ETH Zurich together with Integral Design-Build AG and Halter Group AG. A collaborating author was supported by the Czech Science Foundation (GACR) EXPRO (grant no. 23-07973X). The authors thank Dr. Roxanne Goldberg for editing support.

References

Blank, J. and Deb, K. (2020). pymoo: Multi-objective optimization in python. IEEE Access, 8:89497–89509.

Brinkhoff, T., Kriegel, H.-P., Schneider, R., and Braun, A. (1995). Measuring the Complexity of Polygonal Objects. In ACM-GIS, volume 109.

Casas Arredondo, J. M. (2021). Circular economy and office fit-out: an analysis for office fit-out processes based on material flows. Doctoral, UCL (University College London).

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algo-

rithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197.

Du, C., Esser, S., Nousias, S., and Borrmann, A. (2026). Text2BIM: Generating Building Models Using a Large Language Model-Based Multiagent Framework. Journal of Computing in Civil Engineering, 40(2):04025142. Publisher: American Society of Civil Engineers.

Durmišević, E. (2018). Reversible building design. In Designing for the Circular Economy. Routledge. Num Pages: 16.

Feng, W., Zhu, W., Fu, T.-J., Jampani, V., Akula, A., He, X., Basu, S., Wang, X. E., and Wang, W. Y. (2023). LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. In Advances in Neural Information Processing Systems, volume 36, pages 18225–18250. Curran Associates, Inc.

Fonseca, C., Paquete, L., and Lopez-Ibanez, M. (2006). An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. In 2006 IEEE International Conference on Evolutionary Computation, pages 1157–1163.

Forsythe, P. and Wilkinson, S. (2015). Measuring office fit-out changes to determine recurring embodied energy in building life cycle assessment. Facilities, 33(3-4):262–274.

Herthogs, P., Debacker, W., Tunçer, B., Weerdt, Y. D., Temmerman, N. D., Herthogs, P., Debacker, W., Tunçer, B., Weerdt, Y. D., and Temmerman, N. D. (2019). Quantifying the Generality and Adaptability of Building Layouts Using Weighted Graphs: The SAGA Method. Buildings, 9(4).

Ishibuchi, H., Masuda, H., Tanigaki, Y., and Nojima, Y. (2015). Modified Distance Calculation in Generational Distance and Inverted Generational Distance.

Laignel, G., Pozin, N., Geffrier, X., Delevaux, L., Brun, F., and Dolla, B. (2021). Floor plan generation through a mixed constraint programming-genetic optimization approach. Automation in Construction, 123:103491.

Littlefair, G., Dutt, N. S., and Mitra, N. J. (2025). FlairGPT: Repurposing LLMs for Interior Designs. Computer Graphics Forum, 44(2):e70036. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70036.

McCabe, T. (1976). A Complexity Measure. IEEE Transactions on Software Engineering, SE-2(4):308–320.

Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D., and Benjamin, D. (2017). Project discover: an application of generative design for architectural space planning. In Proceedings of the Symposium on Simulation for Architecture and Urban Design, SIMAUD '17, pages 1–8, San Diego, CA, USA. Society for Computer Simulation International.

Nauata, N., Chang, K.-H., Cheng, C.-Y., Mori, G., and Furukawa, Y. (2020). House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, Computer Vision – ECCV 2020, Lecture Notes in Computer Science, pages 162–177, Cham. Springer International Publishing.

Pacquée, R., Mueller, C., and Rinke, M. (2025). Envisioning alternative buildings: Graph tool shows spatial reuse capacity and informs open-ended design interventions. In Structures and Architecture. CRC Press. Num Pages: 8.

Perron, L., Didier, F., and Gay, S. (2023). The cp-sat-lp solver. In Yap, R. H. C., editor, 29th International Conference on Principles and Practice of Constraint Programming (CP 2023), volume 280 of Leibniz International Proceedings in Informatics (LIPIcs), pages 3:1–3:2, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Schmidt-III, R. and Austin, S. (2016). Adaptable Architecture. Routledge, 0 edition.

Shabani, M. A., Hosseini, S., and Furukawa, Y. (2023). HouseDiffusion: Vector Floorplan Generation via a Diffusion Model with Discrete and Continuous Denoising. In 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5466–5475, Vancouver, BC, Canada. IEEE.

Sheijani, S. S., Momenaei, A., and Hassanzade, H. (2024). Automating modular open-plan office layouts with performance-based generative design. Automation in Construction, 167:105692.

Watt, H., Davison, B., Hodgson, P., Kitching, C., and Densley Tingley, D. (2023). What should an adaptable building look like? Resources, Conservation & Recycling Advances, 18:200158.

Yan, S., Wu, C., and Zhang, Y. (2025). Generative design for architectural spatial layouts: a review of technical approaches. Journal of Asian Architecture and Building Engineering, 0(0):1–21. Publisher: Routledge _eprint: https://doi.org/10.1080/13467581.2025.2512235.

Zhang, H., Savov, A., and Dillenburger, B. (2024a). MaskPLAN: Masked Generative Layout Planning from Partial Input. pages 8964–8973.

Zhang, Z., Guo, Z., Zheng, H., Li, Z., and Yuan, P. F. (2024b). Automated architectural spatial composition via multi-agent deep reinforcement learning for building renovation. Automation in Construction, 167:105702.