2D Floor Plan Generation Via Constraint Satisfaction Problem Modeling

Siddharth[1], Rajgire Akhil Reddy[2], Utkarsh Gupta[2], and Avantika Singh[2]

[1] Department of Computer Science and Engineering,
Dr. Shyama Prasad Mukherjee International Institute of Information Technology,
Naya Raipur, India
siddharth24102@iiitnr.edu.in
[2] Department of Data Science and Artificial Intelligence,
Dr. Shyama Prasad Mukherjee International Institute of Information Technology,
Naya Raipur, India
{rajgire24102,utkarsh24102,avantika}@iiitnr.edu.in

Abstract. Floor plan design is a fundamental architectural task requiring simultaneous satisfaction of multiple spatial, geometric, and functional constraints. Traditional manual design is time-consuming and relies heavily on expert knowledge, while existing automated approaches using deep learning require extensive training datasets and lack interpretability. This paper presents a novel constraint-based approach for automated 2D floor plan generation using Constraint Satisfaction Problem (CSP) formulation solved via Google OR-Tools CP-SAT optimizer. Our framework explicitly models room specifications, adjacency requirements, and geometric constraints as integer programming variables, ensuring satisfaction of hard constraints while optimizing soft constraints. Unlike data-driven methods, our approach requires no training data and produces deterministic, interpretable solutions. The system achieves complete hard constraint satisfaction, excellent space utilization, and high computational efficiency with solve times suitable for interactive architectural design applications. This makes it practical for real-world deployment.

Keywords: Constraint Satisfaction Problem · CP-SAT · Floor Plan Generation · Spatial Optimization

1 Introduction

Floor plan design is a key element in architectural design, real estate development and design of structures [8, 9]. This includes the arrangement of spaces in a building which at the same time meets many conflicting requirements of area constraints, adjacency, boundary inclusion, and non-overlapping placement. Traditional methods depend on expert architects' design, which is a time intensive process, also expensive and difficult to iterate for design alternatives.

Floor plan generation is computationally challenging as it is an NP-complete problem. Constraint satisfaction with spatial overlap involves multi-constraint

optimization that simultaneously addresses area, adjacency, boundary, and non-overlapping requirements. This leads to conflicting objectives requiring trade-offs between space utilization and design quality, along with a combinatorial explosion in the search space for room placements within given boundaries.

Traditional architectural design is at present quite a challenge. Iterative design and revision processes require large architect input, while manual methods are used for rapid prototyping of design options. Recent inroads in AI have brought about automated floor planner methods which use deep learning architectures like Generative Adversarial Nets (GANs), Graph Neural Nets (GNNs), and Transformers [8, 9, 12]. While these research efforts present very impressive visual results, they do have limitations. They require large labeled datasets for training which is computationally expensive. Also they lack interpretability in decision making which is black box in nature, which at times is difficult to trace or identify in the generation process. Also they do not provide guarantees in meeting constraints. Additionally, they struggle with incorporating domain-specific requirements without having to retrain the entire model for different domains, limiting their flexibility and adaptability. Furthermore they require great training overhead in the range of 2 to 5 days of GPU training time [9,12], which in turn makes iterative improvements expensive and impractical for real-world deployment. Despite advances in both learning-based and constraint-based methods, no recent work explores modern constraint programming solvers (CP-SAT) for floor plan generation. Here, in this work we present a novel constraint-based approach for automated 2D floor plan generation using CSP formulation solved via Google OR-Tools CP-SAT optimizer.

The major contributions of this work include three key aspects. First, we present a novel CSP formulation for 2D floor plan generation that explicitly incorporates spatial, geometric, and adjacency constraints as integer programming variables with efficient constraint propagation mechanisms. Second, to the best of our knowledge, this is the first work that utilizes the Google OR-Tools CP-SAT optimizer for automatic floor plan generation, demonstrating the effectiveness of modern constraint programming solvers for architectural layout problems. Third, we conduct a thorough experimental analysis to validate the proposed framework against traditional optimization baseline techniques including random placement, greedy algorithms, and integer linear programming approaches, showing superior performance in constraint satisfaction, space utilization, and computational efficiency. The rest of the paper is organized as follows. In Section 2 we present a comprehensive review of floor plan generation methods. In Section 3, we present the problem formulation and constraint definition. In Section 4, we present experimental results and performance analysis. Section 5 concludes our paper with future directives.

2 Literature Review

Automated floor plan generation has evolved from early constraint-based approaches to modern deep learning architectures. Recent research predominantly

focuses on data-driven methods using neural networks, but these face challenges in interpretability and constraint guarantees. This section reviews key contributions across both paradigms [10, 11] to contextualize our proposed CP-SAT framework.

Table 1 presents a comprehensive analysis of selected representative works in floor plan generation. Several important trends are evident. There is a notable shift toward data-driven approaches, with increasing adoption of deep learning methods over traditional rule-based techniques. Graph representations have gained prominence, with growing use of graph neural networks to model spatial relationships and room connectivity. Generative models, particularly GANs and Transformers, have become the focus for producing realistic floor plan outputs with improved visual quality [8,9,12]. Additionally, the field demonstrates heavy reliance on large labeled datasets, with most contemporary methods requiring extensive floor plan collections for training and validation.

Based on the aforementioned literature we identify three key research issues in present floor plan generation approaches. Present approaches achieve in the range of 75–88% [5, 9, 12] constraint satisfaction, leaving gaps in feasibility which make them unsuitable for professional use where compliance is mandatory. A framework which delivers 100% on critical constraints is necessary in the real world for architecture. Our proposed framework addresses these gaps by providing interpretable constraint reasoning, zero-shot generation ability, and guaranteed satisfaction of hard constraints.

3 Proposed Methodology

This section discusses the proposed framework in detail. Figure 1 represents the generic pipeline of the proposed framework.

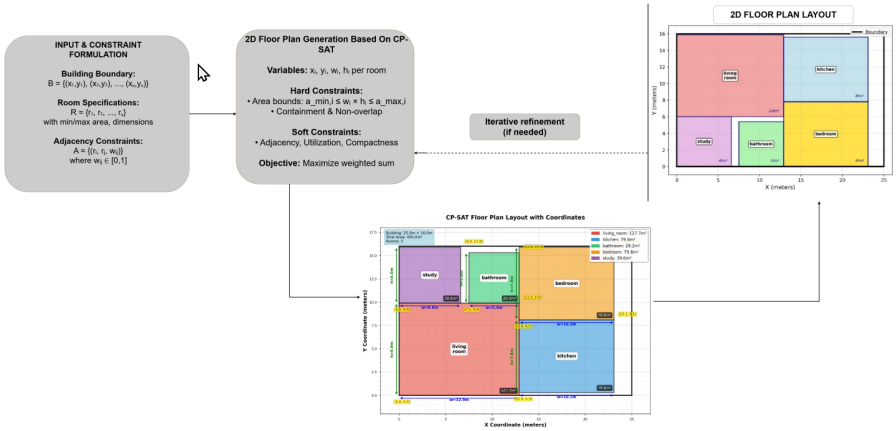


Fig. 1. Generic Pipeline of Proposed Framework.

Table 1. Summary of key floor plan generation approaches.

Author	Proposed Methodology	Remarks (Limitations)
Mostafa et al. (2024) [7]	PARSAC framework using Constraints-Aware Simulated Annealing (CA-SA) for SoC floorplanning with hard placement constraints. Massively parallel approach constructing Pareto-optimal legal solutions.	Designed for chip/SoC floorplanning, not architectural buildings; requires parallel computing infrastructure; computationally intensive.
Zeng et al. (2023) [12]	FloorplanDiffusion model with two-stage generation: room layout generation followed by walls/windows/doors placement using diffusion-based deep learning.	High computational resource requirements; limited to 90° corners; requires substantial training datasets.
Wu et al. (2019) [9]	Two-stage data-driven neural network: room placement using living-room-first strategy, then wall refinement via encoder-decoder architecture trained on RPLAN dataset.	Requires 80K+ labeled floor plans; fixed constraint structures; limited interpretability and flexibility.
Galle (1981) [2]	Exhaustive tree search algorithm generating all rectangular plans on modular grids using backtrack programming with area, adjacency, and orientation constraints.	Limited to rectangular grids; exhaustive search impractical beyond 10 rooms; computationally expensive on legacy hardware.
Charman (1998) [1]	Constraint satisfaction approach using semi-geometric arc-consistency to generate feasible layouts satisfying non-overlap, adjacency, and area requirements.	Computationally heavy for large problems; limited scalability; coarse grid discretization.

3.1 Input and Constraint Formulation

The input specifications consist of three main components. The building boundary $\mathcal{B}$ is defined as:

$$\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \tag{1}$$

where (x_i, y_i) represent the Cartesian coordinates of the i-th vertex, n denotes the total number of vertices, and these vertices collectively define a polygon boundary with total area A_{boundary}. The room specifications $\mathcal{R}$ are represented as:

$$\mathcal{R} = \{r_1, r_2, \ldots, r_m\} \tag{2}$$

where m represents the total number of rooms to be placed within the building boundary. For each room r_i, we define a unique identifier id_i, functional category

type_i (such as living, kitchen, bedroom, etc.), minimum and maximum area bounds ($a_{\min,i}$ and $a_{\max,i}$ in m^2), and minimum width and height dimensions ($w_{\min,i}$ and $h_{\min,i}$ in m). The adjacency constraints $\mathcal{A}$ are specified as:

$$\mathcal{A} = \{(r_i, r_j, w_{ij}) \mid r_i, r_j \in \mathcal{R},\ w_{ij} \in [0, 1]\} \tag{3}$$

where w_{ij} indicates the adjacency priority between rooms i and j, with higher values representing stronger adjacency preferences.

Constraint Categories. The constraints are divided into two categories: *hard constraints* that must be satisfied and *soft constraints* that should be optimized.

Hard constraints include *area bounds*, which ensure that each room's area falls within specified limits:

$$a_{\min,i} \leq w_i \times h_i \leq a_{\max,i}, \quad \forall r_i \in \mathcal{R} \tag{4}$$

Boundary containment constraints ensure all rooms remain within the building boundary:

$$(x_i, y_i) \in \mathcal{B} \ \text{ and } \ (x_i + w_i,\ y_i + h_i) \in \mathcal{B} \tag{5}$$

Non-overlapping constraints prevent spatial conflicts between rooms:

$$\text{overlap}(r_i, r_j) = \emptyset, \quad \forall\, i \neq j \tag{6}$$

Minimum dimension constraints maintain functional room sizes:

$$w_i \geq w_{\min,i} \ \text{ and } \ h_i \geq h_{\min,i} \tag{7}$$

Soft constraints focus on optimizing adjacency satisfaction for high-weight room pairs, maximizing space utilization, maintaining room compactness with aspect ratios near 1, and respecting semantic placement preferences for functional room grouping.

Output Requirements. For each room r_i, the system determines the position through (x_i, y_i) coordinates and the dimensions through (w_i, h_i) width and height values, ensuring all constraints are satisfied while optimizing the overall layout quality.

Optimization Objective. The system maximizes a composite quality score defined as:

$$Q = \alpha \cdot \text{CSR} + \beta \cdot \text{SUE} + \gamma \cdot \text{ASS} + \delta \cdot C \tag{8}$$

where CSR represents the Constraint Satisfaction Rate, SUE denotes Space Utilization Efficiency, ASS indicates the Adjacency Satisfaction Score, and C measures room Compactness. The weight parameters α, β, γ, and δ sum to 1 and determine the relative importance of each metric. This optimization is subject to the requirement that all hard constraints must be satisfied.

Algorithm 1 CP-SAT Floor Plan Generation

Require: $\mathcal{B}$, $\mathcal{R}$, $\mathcal{A}$
Ensure: Floor plan layout L
Validate inputs and compute A_{boundary}
Construct room graph $G = (V, E)$
Initialize CP model $\mathcal{M}$
for each room $r_i \in \mathcal{R}$ **do**
    Create decision variables (x_i, y_i, w_i, h_i)
    Add area constraints
    Add boundary constraints
end for
for each pair (r_i, r_j) **do**
    Add NoOverlap2D constraint
end for
for each $(r_i, r_j, w_{ij}) \in \mathcal{A}$ where $w_{ij} > 0.3$ **do**
    Add adjacency constraint with bonus
end for
Formulate objective function
status $\leftarrow$ Solve($\mathcal{M}$, time_limit = 30s)
if status $\in$ {OPTIMAL, FEASIBLE} **then**
    $L \leftarrow$ ExtractSolution($\mathcal{M}$)
    Validate(L)
    return L
else
    return FallbackPlacement($G, \mathcal{B}$)
end if

3.2 2D Floor Plan Generation Based on CP-SAT

Algorithm 1 presents a step-by-step 2D floor plan generation procedure. The section below gives a detailed overview of this algorithm.

Spatial Discretization. We discretize the continuous space into a grid with resolution $\Delta = 0.3$ metres. The grid dimensions are calculated as:

$$\text{Grid Width} = \lceil (x_{\max} - x_{\min})/\Delta \rceil \tag{9}$$

$$\text{Grid Height} = \lceil (y_{\max} - y_{\min})/\Delta \rceil \tag{10}$$

This discretization allows us to represent the problem as an integer programming formulation while maintaining sufficient spatial resolution for practical architectural applications.

Decision Variables. For each room r_i, we define integer variables representing position and dimensions. The position variables x_i and y_i represent the room's location within the grid, while w_i and h_i represent the room's width and height

respectively:

$$x_i \in [0,\ \text{Grid Width}] \tag{11}$$
$$y_i \in [0,\ \text{Grid Height}] \tag{12}$$
$$w_i \in [\lceil w_{\min,i}/\Delta \rceil,\ \lfloor w_{\max,i}/\Delta \rfloor] \tag{13}$$
$$h_i \in [\lceil h_{\min,i}/\Delta \rceil,\ \lfloor h_{\max,i}/\Delta \rfloor] \tag{14}$$

These variables are bounded by the grid dimensions and the minimum and maximum room size requirements specified in the input.

Constraint Encoding. We encode four types of constraints in the CP-SAT model. *Area constraints* ensure that each room's area falls within specified bounds by using multiplication equality:

$$\texttt{model.AddMultiplicationEquality}(\text{area}_i,\ [w_i, h_i]) \tag{15}$$
$$a^{\text{grid}}_{\min,i} \le \text{area}_i \le a^{\text{grid}}_{\max,i} \tag{16}$$

Boundary constraints ensure all rooms remain within the building boundary:

$$x_i + w_i \le \text{Grid Width} \tag{17}$$
$$y_i + h_i \le \text{Grid Height} \tag{18}$$
$$x_i \ge 0, \quad y_i \ge 0 \tag{19}$$

Non-overlap constraints prevent spatial conflicts between rooms. For each room pair (r_i, r_j), we create 2D interval variables:

$$X_i = \text{IntervalVar}(x_i,\ w_i,\ x_i + w_i) \tag{20}$$
$$Y_i = \text{IntervalVar}(y_i,\ h_i,\ y_i + h_i) \tag{21}$$

and enforce the non-overlap constraint:

$$\texttt{model.AddNoOverlap2D}([X_i, X_j], [Y_i, Y_j]) \tag{22}$$

Adjacency constraints promote desired spatial relationships between rooms. For high-weight pairs where $w_{ij} > 0.7$, we create boolean variables indicating whether rooms share edges. The vertical adjacency condition is formulated as:

$$(x_i + w_i = x_j)\ \wedge\ (y_i < y_j + h_j)\ \wedge\ (y_j < y_i + h_i) \tag{23}$$

ensuring that rooms are adjacent along their vertical boundaries with overlapping heights.

3.3 Room Graph Representation

We model spatial relationships using a graph $G = (V, E)$ where:

- Nodes V represent rooms with attributes: $\{\text{type},\ a_{\min},\ a_{\max}\}$
- Edges E represent adjacency constraints with weights w_{ij}

The adjacency matrix M is constructed as:

$$M_{ij} = \begin{cases} w_{ij} & \text{if } (r_i, r_j) \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

Table 2. Example Room Specifications (Sample Input).

Room	Type	Min Area	Max Area
Living Room	living	50 m^2	80 m^2
Kitchen	kitchen	20 m^2	40 m^2
Bedroom	bedroom	30 m^2	60 m^2
Study	study	12 m^2	25 m^2
Bathroom	bathroom	8 m^2	15 m^2
Total	—	120 m^2	220 m^2

3.4 2D Floor Plan Layout

The output of the CP-SAT algorithm is given as an input to the NetworkX [3] library. Afterwards, with the help of the matplotlib [4] library, 2D floor plan images with intuitive graphical depictions are generated for architectural review and client presentation.

4 Experimental Results and Discussion

In the following section, experimental setup, evaluation metrics, and experimental results are presented.

4.1 Experimental Setup

The system was implemented using Python 3.8+ as the main programming language, which also uses Google OR-Tools CP-SAT v9.x as the solver [6] and NetworkX 2.6+ for graph representation. We used a standard workstation equipped with an Intel i7-8550U running at 1.8 GHz and 16 GB of RAM operating on Ubuntu 20.04 LTS.

The experimental analysis was done on a rectangular building footprint that measures 25 m × 16 m, totaling 400 m^2. The test consisted of 5 rooms which included a living room, kitchen, bedroom, bathroom, and home office, representing a typical residential layout. For spatial discretization, we used a grid resolution of 0.3 metres, which provided sufficient granularity for architectural design while maintaining computational efficiency. The CP-SAT solver was set with a limit of 30 seconds per run to assess performance with regard to interactive design requirements. To ensure statistical validity, we conducted 10 independent executions of each test setup, which allows us to analyze consistency and variability in solution quality and performance. Table 2 and Table 3 depict example room specifications and example adjacency requirements of our proposed approach.

4.2 Evaluation Metrics

The generated floor plans are evaluated using a comprehensive set of quantitative constraint-based metrics that assess both geometric accuracy and semantic

Table 3. Example Adjacency Requirements (Sample Input).

Room Pair	Weight	Priority
Living ↔ Kitchen	0.9	Critical
Bedroom ↔ Bathroom	0.8	High
Living ↔ Study	0.6	Medium
Kitchen ↔ Bathroom	0.5	Medium
Living ↔ Bedroom	0.4	Low
Study ↔ Bedroom	0.3	Low

quality. The metrics reported in this paper are the Hard Constraint Satisfaction Rate (HCSR), Space Utilization Efficiency (SUE), Room Compactness Score (RCS), and Adjacency Satisfaction Score (ASS).

1. **Hard Constraint Satisfaction Rate (HCSR):**

$$\text{HCSR} = \frac{\text{Satisfied Hard Constraints}}{\text{Total Hard Constraints}} \times 100\% \tag{25}$$

2. **Space Utilization Efficiency (SUE):**

$$\text{SUE} = \frac{\sum_{i=1}^{n}(w_i \times h_i)}{A_{\text{boundary}}} \times 100\% \tag{26}$$

3. **Room Compactness Score (RCS):**

$$\text{RCS} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \left|\frac{\max(w_i, h_i)}{\min(w_i, h_i)} - 1.5\right| \cdot 0.5\right) \times 100\% \tag{27}$$

4. **Adjacency Satisfaction Score (ASS):**

$$\text{ASS} = \frac{\sum_{(i,j)\in\mathcal{A}} w_{ij} \cdot I_{\text{adj}}(i,j)}{\sum_{(i,j)\in\mathcal{A}} w_{ij}} \times 100\% \tag{28}$$

where $I_{\text{adj}}(i,j) = 1$ if rooms i and j share a wall, 0 otherwise.

The composite **Layout Quality Index (LQI)** is:

$$\text{LQI} = \alpha \cdot \text{HCSR} + \beta \cdot \text{SUE} + \gamma \cdot \text{ASS} + \delta \cdot \text{RCS} \tag{29}$$

where $\alpha = 0.30$, $\beta = 0.25$, $\gamma = 0.30$, and $\delta = 0.15$.

Table 4. Comprehensive Evaluation Metrics for Generated Floor Plan.

Metric	Score	Performance
Hard Constraint Satisfaction (HCSR)	100%	Excellent
Space Utilization Efficiency (SUE)	85.82%	Very Good
Room Compactness Score (RCS)	90%	Excellent
Adjacency Satisfaction Score (ASS)	81.40%	Good
Layout Quality Index (LQI)	**89.38%**	**Very Good**

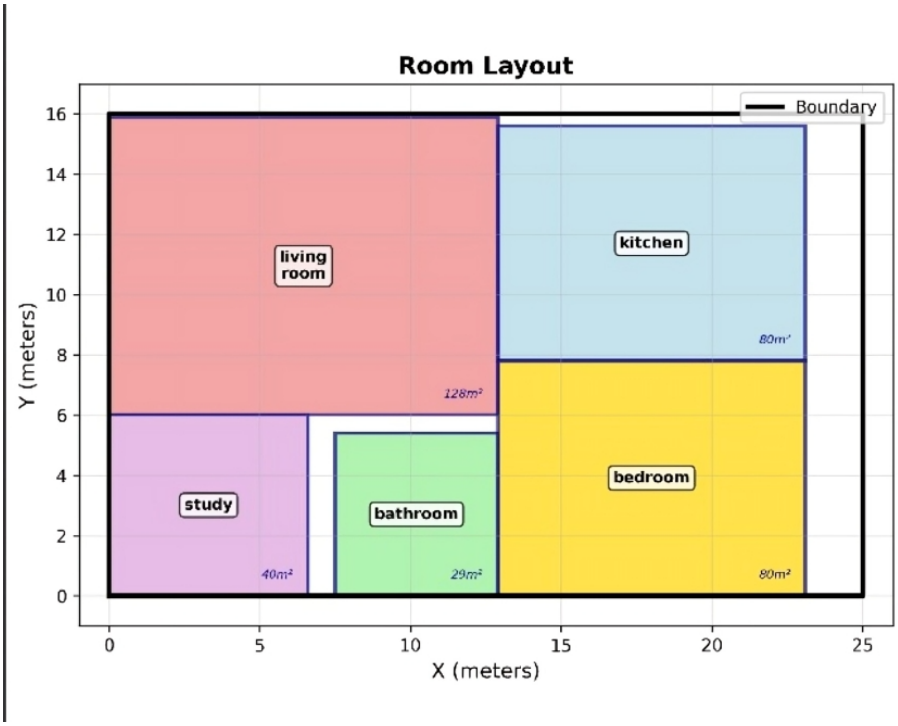


Fig. 2. Generated floor plan layout showing optimal room arrangement with hard constraint satisfaction (HCSR = 100%) and high space utilization (SUE = 85.82%).

4.3 Experimental Results

Figure 2 depicts the generated floor plan layout of our proposed framework. Table 4 presents the comprehensive evaluation metrics for the generated 2D floor plan.

Key Findings. All hard constraints were met by the generated layout, guaranteeing that rooms fit precisely inside boundaries without overlapping. High space

utilization indicates minimal waste and effective use of the available area. Effective spatial relationships are reflected in strong adjacency performance, while well-shaped, functional rooms are indicated by the compactness measure. All things considered, the arrangement exhibits a well-balanced, effective, and cohesive design.

5 Conclusion and Future Directions

This study introduced a novel constraint-based method for automatically creating 2D floor plans through the formulation of Constraint Satisfaction Problems and CP-SAT optimization. Our framework guarantees the fulfillment of crucial requirements while maximizing layout quality by explicitly modeling spatial, geometric, and adjacency constraints as integer programming variables. Experimental validation on a 400 m^2 building with 5 rooms demonstrated complete hard constraint satisfaction at 100%, excellent space utilization at 85.82%, good adjacency satisfaction at 81.40%, and an overall layout quality index of 89.38% with average solve times under 8 seconds. Our method fills important research gaps in floor plan generation by offering a workable, dependable, and understandable substitute for data-driven deep learning techniques. The main benefits include zero-shot generation that does not require any training data, guaranteed feasibility through complete hard constraint satisfaction, interpretable reasoning with explicit constraint handling, and computational efficiency appropriate for interactive applications. The system exhibits preparedness for implementation in practical architectural design support applications.

Future research will concentrate on the amalgamation of deep learning models with the CSP framework to improve initialization and optimize the fulfillment of aesthetic and soft constraints. Adding more types of constraints, like lighting, ventilation, accessibility, and regulatory requirements, will make the model more useful in the real world. Creating an interactive interface with real-time constraint editing, 3D visualization, and CAD export will make architectural workflows run smoothly. Adding commercial, healthcare, educational, and industrial layouts to the system will show how flexible and useful it can be in a business setting.

References

1. Charman, P.: A constraint-based approach for the generation of floor plans. In: Proceedings of the SECOIA Project, CERMICS-INRIA. Sophia Antipolis, France (1998)
2. Galle, P.: An algorithm for exhaustive generation of building floor plans. Environment and Planning B: Planning and Design **8**(1), 87–103 (1981). `https://doi.org/10.1068/b080087`
3. Hagberg, A.A., Swart, P.J., Schult, D.A.: Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference. pp. 11–15 (2008)

4. Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
5. Laignel, G., Pozin, N., Geffrier, X., Delevaux, L., Brun, F., Dolla, B.: Floor plan generation through a mixed constraint programming–genetic optimization approach. Automation in Construction **142**, 104509 (2022). https://doi.org/10.1016/j.autcon.2022.104509
6. Moreira, E.J.B., de Freitas, S.A.A.: A CP-SAT approach for academic resource timetabling in higher education institutions: A case study at a major public university. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 1–6 (2023). https://doi.org/10.1109/SMC53992.2023.10347651
7. Mostafa, H., Mallappa, U., Galkin, M., Phielipp, M., Majumdar, S.: PARSAC: Fast, human-quality floorplanning for modern SoCs with complex design constraints. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–9 (2021). https://doi.org/10.1109/ICCAD51958.2021.9643491
8. Saliu, N., Elezi, K.: Integration of artificial intelligence in floorplan layout planning: A systematic review of models, methods, and applications. Journal of Architectural Design and Technology **12**(2), 45–68 (2025)
9. Wu, W., Fu, X.M., Tang, R., Wang, Y., Qi, Y.H., Liu, L.: Data-driven interior plan generation for residential buildings. ACM Transactions on Graphics (TOG) **38**(6), 1–12 (2019). https://doi.org/10.1145/3355089.3356556, article 234
10. Xu, Y.W., Li, W., Sanner, S., Khalil, E.B.: Self-supervised transformers as iterative solution improvers for constraint satisfaction. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
11. Zawidzki, M., Tateyama, K., Nishikawa, I.: The constraint satisfaction problem approach in the design of an architectural functional layout. Architectural Engineering and Design Management **6**(4), 227–241 (2010). https://doi.org/10.3763/aedm.2010.0122
12. Zeng, P., Gao, W., Yin, J., Xu, P., Lu, S.: Residential floor plans: Multi-conditional automatic generation using diffusion models. Automation in Construction **155**, 105040 (2023). https://doi.org/10.1016/j.autcon.2023.105040