Constrained Layout Generation with Factor Graphs

Mohammed Haroon Dupty[1,*] Yanfei Dong[1,2] Sicong Leng[3] Guoji Fu[1]
Yong Liang Goh[1] Wei Lu[3] Wee Sun Lee [1]
[1] National University of Singapore [2] Paypal [3] Singapore University of Technology and Design
{haroon, dyanfei, guoji.fu, gyl}@u.nus.edu, sicong_leng@mymail.sutd.edu.sg,
luwei@sutd.edu.sg, leews@comp.nus.edu.sg

Abstract

This paper addresses the challenge of object-centric layout generation under spatial constraints, seen in multiple domains including floorplan design process. The design process typically involves specifying a set of spatial constraints that include object attributes like size and inter-object relations such as relative positioning. Existing works, which typically represent objects as single nodes, lack the granularity to accurately model complex interactions between objects. For instance, often only certain parts of an object, like a room's right wall, interact with adjacent objects. To address this gap, we introduce a factor graph based approach with four latent variable nodes for each room, and a factor node for each constraint. The factor nodes represent dependencies among the variables to which they are connected, effectively capturing constraints that are potentially of a higher order. We then develop message-passing on the bipartite graph, forming a factor graph neural network that is trained to produce a floorplan that aligns with the desired requirements. Our approach is simple and generates layouts faithful to the user requirements, demonstrated by a large improvement in IOU scores over existing methods. Additionally, our approach, being inferential and accurate, is well-suited to the practical human-in-the-loop design process where specifications evolve iteratively, offering a practical and powerful tool for AI-guided design.

1. Introduction

The incorporation of AI into the layout design process represents a significant advancement in design methodologies. Researchers have predominantly adopted a generative modeling approach, enabling the generation of diverse designs with limited user inputs [1, 15, 17, 20, 21]. Although very useful, this approach faces practical challenges when we have a fixed uneven boundary and users have several specific requirements and seek active involvement in the design process. In such instances, inferential models specifically trained to produce layouts that align precisely with user requirements are likely more beneficial.

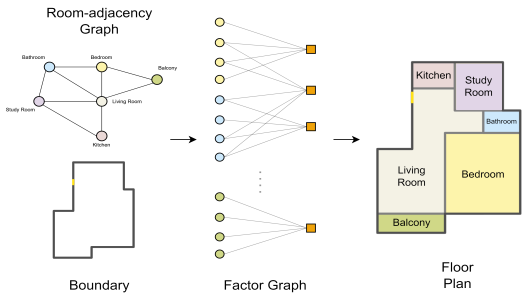

Figure 1. Given a building boundary and a graph that encodes user requirements and room constraints, we first transform the inputs into a factor graph that can model higher-order constraints, and then produce the floorplan based on the factor graph model.

The challenge of layout design generation with constraints is more prominently seen in the design of *floorplan layouts* [1, 11, 14, 15, 17, 20, 21, 23, 31, 33], which can include both user-defined and structural constraints. User-defined constraints stem from individual preferences regarding room sizes, adjacencies, and other design considerations. Structural constraints may require the layout to remain within the boundary of a building or adhere to other geometric restrictions. For example, given a boundary, a user may want two rooms adjacent to each other in a particular corner of the house. A good inferential model would be able to produce a layout satisfying these constraints. With partial output available, users can append or update their requirements and thereby, refine the design iteratively.

One common approach for producing floorplan layouts is to use a two-stage method, where the first stage involves predicting bounding boxes for individual objects/rooms, followed by a refinement network that converts the bounding boxes into an image [4, 11]. For the first stage, the room-adjacency graph is taken as input, and a Graph Neural Network(GNN)-based model is often used to predict the

*Corresponding Author

bounding boxes of rooms [5, 11]. However, the representation of each object as a single node, while straightforward, falls short in terms of granularity and effectiveness. It notably struggles to accurately reflect the intricate interactions that occur between objects, as these interactions are frequently localized to specific parts of an object. For example, it is often the case that only one side of a room, such as the right wall, is actively engaged with an adjacent object. Therefore, we represent each room using four separate latent variables to represent the four boundaries of the bounding box: x_{min}, x_{max}, y_{min}, and y_{max}. This approach effectively captures fine-grained interactions, yet it also introduces the necessity to model higher-order dependencies. For instance, determining the size of a room requires all four variables. Modeling the potentially higher-order dependencies, however, presents a non-trivial challenge.

To handle the challenge, we propose a factor graph based model to encode the dependencies. A factor graph is a bipartite graph consisting of factor nodes and variable nodes. In our model, each room is represented by four variable nodes, and each factor node is designed to represent the dependencies among its connected variables. This enables modeling arbitrary constraints including those of a higher-order, a capability lacking in traditional GNNs which can model only pairwise dependencies. This allows us to leverage domain knowledge more effectively than the adjacency-graph approach typically used in existing works. For instance, we impose a boundary constraint ensuring that all rooms are contained within a predefined boundary by employing a factor that connects to all the relevant variables. We associate an embedding with each factor graph node to represent the latent information associated with the node and design message passing operations on the factor graph, leading to a Factor Graph Neural Network (FGNN) [8, 36].

As our objective is to produce floorplan layouts that satisfy all input constraints, we evaluate our model's performance with a focus on fidelity to the ground-truth image that is used to derive the constraints. Our model surpasses other strong baselines, achieving a significant improvement in both box-level IOUs and pixel-level metrics, indicating the effectiveness of our approach. Furthermore, we demonstrate our model's usefulness in two distinct practically useful scenarios. One, we show that our model is well-suited to the iterative refinement of human requirements, enabling the development of varied designs with user-interactions. Two, we show that our model can form part of a generative pipeline and generate diverse realistic layout designs for the same given input boundary, validated both qualitatively and quantitatively.

2. Related Work

The predominant approach taken by the researchers towards addressing the automated design problem is through conditional generative modeling [1, 11, 12, 14, 15, 17, 20, 21, 23, 31, 33]. Layout generative models with a focus on residential floorplan layouts, aim to create room layouts that satisfy limited user constraints of room types and adjacencies, with or without building boundaries [2, 6, 33, 34, 38]. Previous works have used various strategies to achieve this, such as training probabilistic methods [19, 27] or using evolutionary strategies based on user specifications and constraints [25, 26], and designing constraint graphs [11, 23].

In[33], a two-stage approach learning network was proposed for generating floorplans for residential buildings, and the RPLAN dataset was released. Since then, follow-up works have focused mainly on building GAN-based generative models. These models typically take in a room-adjacency graph called bubble diagrams. Some of the popular models in this line of research include Housegan, Housegan++, and other variants [3, 20–23, 29, 31, 32]. While these approaches generate realistic floorplans, they have limitations when users are seeking to actively vet most aspects of the layout. Recently, [9] have proposed a generative modeling approach which allows human-in-the-loop mechanism. However, they do not exploit the relational constraints that users are likely to provide. Furthermore, in this work, we target the problem from an iterative design perspective where users refine their requirements iteratively upon seeing the results of their partial requirements. In such cases, an inferential model which focuses on maximizing fidelity to the user requirements is likely more useful.

Research on floorplan layout generation, focused on creating layouts from input room-adjacency graphs that match ground truth is found in Graph2Plan [11] and HPGM [5]. Our work builds upon Graph2Plan's two-stage methodology, initially predicting room bounding boxes and then utilizing these predictions for layout creation. We propose transforming input constraints into a factor graph and employing neuralized message passing for effective constraint utilization and domain knowledge integration. Neural message passing on factor graphs has recently shown impressive results on some tasks like molecular prediction and navigation [7, 8, 28, 30, 35, 36]. Our work mainly builds on [8, 36, 37] with learnable bipartite message passing between variable and factor nodes.

3. Preliminaries

3.1. Factor Graphs

A factor graph is a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ with two disjoint sets of nodes, variable nodes $\mathcal{V}$, and factor nodes $\mathcal{C}$. A variable node $i \in \mathcal{V}$ represents a variable $x_i \in \mathbf{x}$ and each factor node $c \in \mathcal{C}$ specifies that there are dependencies among a set of variables $\mathbf{x}_c$; an edge $(c, i) \in \mathcal{E}$ exists between variable node i and factor node c if $x_i \in \mathbf{x}_c$.

Factor graphs are widely used to capture dependencies

in probabilistic graphical models (PGM). In PGMs, approximation algorithms of inferring optimal values of variables $\mathbf{x}^*$ often operate by running message passing on factor graphs, for example, the *belief propagation* algorithms.

3.2. Factor Graph Neural Network

Message-passing operations on a graph can be represented as a graph neural network. In a similar way, we can represent message passing on a factor graph as a factor graph neural network (FGNN) [8, 28, 36]. By assuming tensor decomposed factor potentials, both sum-product and max-product belief propagation algorithms for PGMs can be represented with compact FGNNs [8, 36, 37]. With the use of more powerful approximators within the FGNN, it may be possible to learn even more powerful inference algorithms specialized for the problem domain.

In FGNN, a bipartite factor graph is established with variable and factor nodes, where each factor node connects to its dependent variable subset. These nodes are represented by variable ($\mathbf{v}_i$) and factor ($\mathbf{f}_c$) embeddings, initialized using respective features. The embeddings are then updated iteratively through message-passing equations.

$$\tilde{\mathbf{v}}_i = \text{AGG}_{c \in N(i)} \; \Psi_{\text{FV}}(\mathbf{f}_c, \mathbf{v}_i) \tag{1}$$

$$\tilde{\mathbf{f}}_c = \text{AGG}_{i \in N(c)} \; \Psi_{\text{VF}}(\mathbf{v}_i, \mathbf{f}_c) \tag{2}$$

where Ψ_{FV} and Ψ_{VF} are functions with learnable parameters. Note that the factor graph is a bipartite graph. Therefore, $N(i)$ will contain only factor nodes c and $N(c)$ will contain only variable nodes i. This process can be iterated multiple times, followed by a readout function for tasks like regression or classification.

4. Proposed Method

4.1. Problem Setup

In our problem setup, we are given a room-adjacency graph G and a boundary of the building B. The graph G is as described in Graph2Plan [11] and contains a node for each room with attributes including the size and position, and edges are typed adjacencies describing the spatial relationship between rooms. The boundary B is a 2D binary mask describing the inside area of the building. The task is to produce the floorplan layout image $\mathcal{I}$ which satisfies all the constraints in the input. Our problem setup differs from many GAN-based methods, as they often prioritize generation diversity. In contrast, our objective is to produce a layout that satisfies the specified constraints. Therefore, in addition to generating realistic layouts, we also aim to produce layouts that closely resemble the ground truth.

To address the task, we follow conventional methods in adopting a two-stage approach [5, 11]. In the first stage, we output the coordinates of bounding boxes for each room, while in the second stage, we produce the floor plan layout image using the predicted bounding boxes. Our primary focus is on enhancing the accuracy of the first stage by leveraging domain knowledge, such as higher-order relations. To this end, we seek to effectively incorporate domain knowledge, including higher-order relations, into the model.

4.2. Representation

For the first stage of predicting bounding boxes, conventional methods leverage GNNs to process the input graph and output bounding box coordinates, which are then used to produce the layout image. However, representing each object as a single node is a straightforward approach but lacks in granularity and effectiveness. This method often fails to capture the complex interactions between objects, which are typically concentrated on certain parts of an object. A common example is that only a specific side of a room, like the right wall, might be actively interacting with a neighboring object.

To capture the fine-grained interactions, we propose to represent each room with four different variables. Consider a floorplan image as a single-channel image of size $H \times W$ with each room represented by an enclosing bounding box. Let the room i's bounding box be described by four coordinates in the form $(x^i_{min}, x^i_{max}, y^i_{min}, y^i_{max})$ with $x_{min/max} \in \{1 \dots W\}$ and $y_{min/max} \in \{1 \dots H\}$. Then the set of variables $\mathbf{x}$, describing the bounding boxes of all the rooms, can be stated as

$$\mathbf{x} = \left\{ x^i_{min}, x^i_{max}, y^i_{min}, y^i_{max} |\; i \in \{1 \dots N\} \right\}$$

where N is the number of rooms in a given floorplan. Given the boundary B and the input graph G, we aim to predict the values of the configuration of variables $\mathbf{x}$ which optimally satisfies all the constraints in B and G.

4.3. Floor-Plan Factor Graph Model

In layout design, we frequently encounter high-order constraints that traditional GNNs cannot capture due to their limitation in modeling only pairwise dependencies. To effectively encode these constraints, we propose the adoption of factor graphs. Factor graphs provide the needed modeling capacity to capture the information available from domain knowledge. With factor graphs, the mutual dependencies between the variables can be easily encoded in the form of factor nodes. Importantly, input constraints provide important clues regarding the variable dependencies. For example, with our defined variables $x_{min}, x_{max}, y_{min}, y_{max}$ per each box, when two rooms are adjacent with a *left-right* relationship, it is likely that the x_{max} and x_{min} values of the two rooms are correlated. By recognizing these correlations, we can leverage domain knowledge to enhance the model's accuracy in predicting bounding box coordinates.

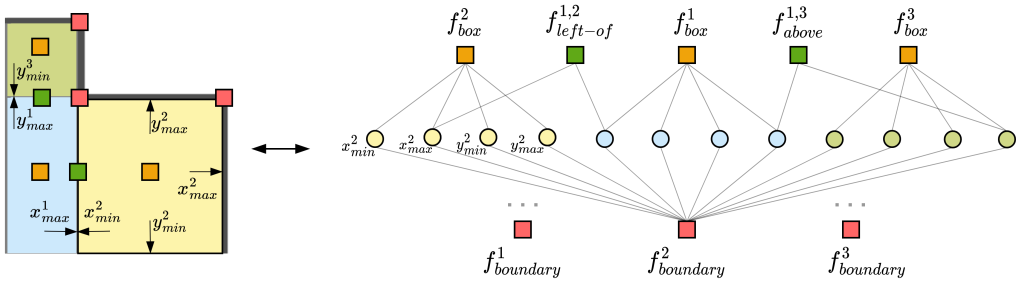

Figure 2. Illustration of the proposed factor graph model for floorplan design. Each room is represented with four bounding-box variables. Factors connect the variables based on input constraints and domain knowledge. f^i_{box} connects four variables of room i. $f^{s,o}_{rel}$ connects only relevant subset of variables between rooms s and o. $f^k_{boundary}$ represents k^{th} corner-point and connects to all the variables. Message passing on this factor graph helps learn better room coordinates which are then used to produce the layout image.

Relation	*left-of*	*right-of*	*above*	*below*
Factors	$\{x^s_{max}, x^o_{min}\}$	$\{x^s_{min}, x^o_{max}\}$	$\{y^s_{max}, y^o_{min}\}$	$\{y^s_{min}, y^o_{max}\}$

Relation	*left-above*	*right-above*	*left-below*	*right-below*
Factors	$\{x^s_{max}, x^o_{min}\}$ $\{y^s_{max}, y^o_{min}\}$	$\{x^s_{min}, x^o_{max}\}$ $\{y^s_{max}, y^o_{min}\}$	$\{x^s_{max}, x^o_{min}\}$ $\{y^s_{min}, y^o_{max}\}$	$\{x^s_{min}, x^o_{max}\}$ $\{y^s_{min}, y^o_{max}\}$

Relation	*inside*	*surrounding*
Factors	$\{x^s_{min}, x^o_{min}\}$ $\{y^s_{min}, y^o_{min}\}$ $\{x^s_{max}, x^o_{max}\}$ $\{y^s_{min}, y^o_{max}\}$	$\{x^s_{min}, x^o_{min}\}$ $\{y^s_{min}, y^o_{min}\}$ $\{x^s_{max}, x^o_{max}\}$ $\{y^s_{max}, y^o_{min}\}$

Table 1. Full set of relation types and their corresponding factors f_{rel} defined in our factor graph model.

Overall, we model four kinds of factors: box factors, relational factors, boundary factors, and a complete factor. Except relational factors, the rest are higher-order factors that capture dependencies between multiple variables. An illustration of the model is shown in Fig. 2.

4.3.1 Bounding Box Factors

We have four variables per bounding box, and these variables are likely to be highly correlated. For example, if the room size is 100 sqft, the four variables of the room must satisfy the constraint that $(x_{max} - x_{min}) * (y_{max} - y_{min}) = 100$. This constraint is a higher-order factor that can provide a useful inductive bias for learning the values of the x and y variables. To encode this information, we introduce a box factor for each room i, denoted as $f^i_{Box} = \{x^i_{max}, x^i_{min}, y^i_{max}, y^i_{min}\}$, into the factor graph model.

4.3.2 Relation factors

The spatial room adjacencies available in the input graph provide rich information about the relative locations of the rooms and need to be effectively exploited. We work on the typed adjacencies defined in Graph2Plan [11], which are based on the spatial relation type of the pair of adjacent rooms. The spatial relation types are: *left-of, right-of, above, below, left-above, left-below, right-above, right-below, inside, and surrounding*.

Our factor-graph-based modeling with 4 variables per room, allows us to only link the variables which are likely to have direct dependencies. We introduce factors connecting specific variables in rooms s and o depending on the relation type p of the typed adjacency $< s, p, o >$. For instance, if the relation is $< room_s, \textit{left-of}, room_o >$, it is probable that the x_{max} coordinate of room s is almost equal to the x_{min} coordinate of room o. Therefore, we add a factor $f^{s,o}_{rel} = \{x^s_{max}, x^o_{min}\}$ to capture this relation.

Similarly, we introduce factors based on the relation types which imply specific coordinate relationships between the subject and the object rooms. Table 1 describes the full set of relational factors in our model for each of the relation types. Overall, we have 20 distinct relational factor types for the 10 types of defined relations.

4.3.3 Boundary factors

In practice, the outer boundary of a floorplan is one of the most important constraints as it directly affects the placement of each room and the way in which different rooms should be aligned within the floorplan boundary. However, incorporating such boundary information is not trivial. Previous methods like RPLAN and Graph2Plan [11, 33] use a CNN encoder with a 2-dimensional binary boundary mask to output a boundary embedding, which is then used to produce the floorplan. However, this process is less effective because only the outer wall structure contains important information, not the whole binary mask.

To address this, we introduce a highly effective method to incorporate boundary information. We observe that the most influential information contained in the boundary is

the corner points of the boundary. The corner points can be described by the (x, y) coordinates and hence provide better inductive bias than a plain binary mask. Therefore, we first extract the set of corner points from the boundary mask. Then, for each corner point, we introduce a higher-order boundary factor that connects to all the variable nodes within the factor graph model. Specifically, let the boundary $b = \{(x, y)_b^k \mid k \in \{1, 2, \dots n_b\}\}$ where n_b is the number of corner points extracted from the given boundary mask. Then for k^{th} corner point, we define a higher-order factor $f_b^k = \{x_{min}^i, x_{max}^i, y_{min}^i, y_{max}^i | i \in \{1 \dots N\}\}$ which connects to all variables in the factor graph.

We encode the information describing the location and surroundings of the corner point as an input feature $\mathbf{f}_b^k$ of the corner-point factor f_b^k. Specifically, for each corner point b^k, a feature vector is formed and initialized with three types of information: the (x, y) coordinates of the corner point, the distance d of the corner point from the bounding box enclosing the boundary, and the binary boundary $mask$ values surrounding the corner point at a small offset of ϵ. The surrounding values indicate the direction of the corner point enclosing the inside of the house. This vector is used as an input feature for each higher-order corner-point factor.

$$\mathbf{f}_b^k = \big[x,\ y,\ d_{left},\ d_{right},\ d_{top},\ d_{bottom},\ mask_{(x+\epsilon,y+\epsilon)},\ mask_{(x+\epsilon,y-\epsilon)},\ mask_{(x-\epsilon,y+\epsilon)},\ mask_{(x-\epsilon,y-\epsilon)}\big]_b^k$$

4.3.4 Complete Factor

In addition to the variable dependencies described above, there can be other dependencies present in the data that are not apparent. In order to enable variables to learn from such hidden dependencies, we introduce an additional factor called the *complete factor*. This factor connects all the variables without any restriction with a factor feature indicating the type of the factor.

4.4. FloorPlan Factor Graph Neural Network

We turn our factor graph model into a neural network with learnable potential functions. Specifically, we initialize our factor graph model with variable node features $[\mathbf{v}_i]_{i\in\mathcal{V}}$ and factor node features $[\mathbf{f}_c]_{c\in\mathcal{C}}$. The variable node feature contains room-type, location, size and a 4-dim one-hot vector indicating the variable type *i.e.* $x_{min}, x_{max}, y_{min}$ or y_{max}. Note that the location here is not the (x, y) coordinate but the approximate relative placement indicating *north, south, north-west etc.*, as discussed in Graph2Plan [11]. The factor features are initialized with the factor type (*i.e.* box factor, relation-type, boundary or complete) and additional specific attributes for some factors (*i.e.* The bounding box factors f_{box}^i contain room-type, location, and size, and the boundary factors f_b^k additionally contain the boundary feature vector as defined in Section 4.3.3.

Algorithm 1 FloorPlan-FGNN (FP-FGNN) Algorithm

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E}), [\mathbf{v}_i]_{i\in\mathcal{V}}, [\mathbf{f}_c]_{c\in\mathcal{C}}, [e_{ci}]_{(c,i)\in\mathcal{E}}$
Output: $[v_i]_{i\in\mathcal{V}}$, $\mathcal{I}$ // Box coordinates and Layout image

for $l = 1, 2, \dots, L$ **do**
  Variable-to-Factor Message Passing:
    $\tilde{\mathbf{f}}_{ci}^l = \text{MLP}_{\text{VF}}\left(\text{Concat}[\mathbf{f}_c^l, \mathbf{v}_i^l, e_{ci}]\right)$
    $\mathbf{f}_c^{l+1} = \underset{i\in N(c)}{\text{softmax}}\left(\tilde{\mathbf{f}}_{ci}^l | \theta_{\text{VF}}^l\right)$
  Factor-to-Variable Message Passing:
    $\tilde{\mathbf{v}}_{ic}^l = \text{MLP}_{\text{FV}}\left(\text{Concat}[\mathbf{f}_c^l, \mathbf{v}_i^l, e_{c,i}]\right)$
    $\mathbf{v}_i^{l+1} = \underset{c\in N(i)}{\text{softmax}}\left(\tilde{\mathbf{v}}_{ic}^l | \theta_{\text{FV}}\right)$
end for
$v_i = \text{MLP}(\mathbf{v}_i^{\text{L}})$ // Train with L1-Loss
$\mathcal{I} = \text{CRN-Network}\big([v_i]_{i\in\mathcal{V}}\big)$ // Pixel-wise cross-entropy

Algorithm 1 details our Factor Graph Neural Network (FP-FGNN) for floorplan design. It takes a Factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$ as input, with variables $\mathcal{V}$ and factors $\mathcal{C}$ initialized using variable features $[\mathbf{v}_i]i \in \mathcal{V}$ and factor features $[\mathbf{f}_c]c \in \mathcal{C}$. The edge set $\mathcal{E}$ contains edges between variable and factor nodes. The edge feature e_{ci} contains the features of factor c.

In each iteration, the variable nodes receive and aggregate messages from the factor nodes and vice versa. The message function is an MLP acting on the concatenated variable and factor embeddings from the previous iteration along with the edge feature. Furthermore, our message aggregation allows weighted aggregation of messages with the help of a learnable softmax-based aggregator defined as

$$\text{softmax}(\mathbf{v}\,|\theta) = \sum_{\mathbf{v}_i\in\mathbf{v}} \frac{\exp(\theta\cdot\mathbf{v}_i)}{\sum_{\mathbf{v}_j\in\mathbf{v}}\exp(\theta\cdot\mathbf{v}_j)}\cdot\mathbf{v}_i \quad (3)$$

This enables variables to learn from factors most relevant to them and vice versa. After a fixed number of iterations, we use an MLP to output the coordinate value of each variable.

4.5. Bounding boxes to floorplan image

To translate the box coordinates to the floorplan image, we feed the predicted bounding box coordinates to a cascaded refinement network (CRN)[4] based network in the form of multi-channel input. We borrow this part of the network from Graph2Plan [11], which can produce the floorplan image given the predicted bounding boxes.

4.6. Training

Our network is trained with only two loss functions. The bounding box coordinate variables are trained with mean absolute loss (L1-Loss) against the ground-truth box-coordinate values and the final layout image is trained with

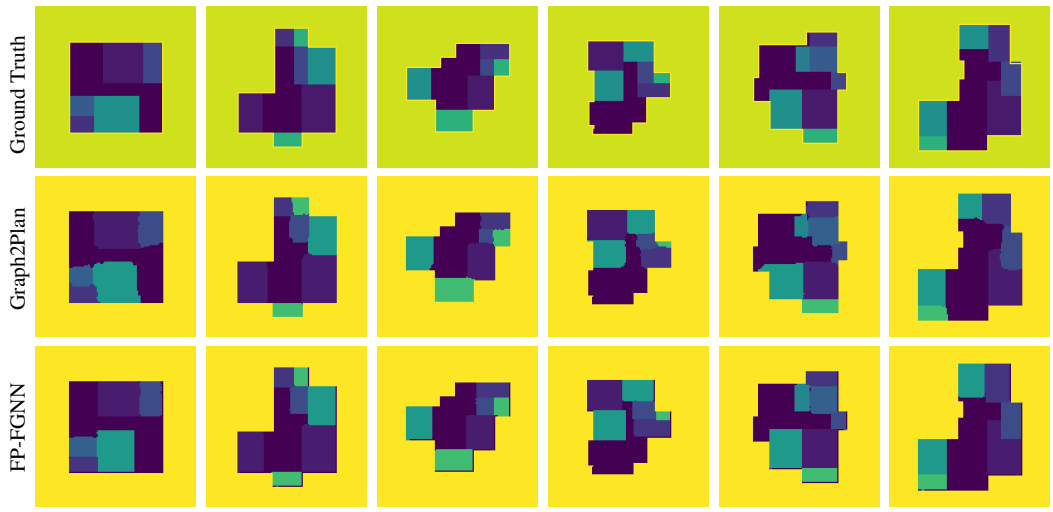

Figure 3. Visualization of layout predictions of Graph2Plan and FP-FGNN without postprocessing. FP-FGNN is able to produce more accurate predictions of room dimensions and the shape of walls, resulting in layouts that are more visually coherent and well-defined.

Method	Box-level		Pixel-level			Constraints-level	
	IOU-Macro	IOU-Micro	Accuracy	IOU-Macro	IOU-Micro	Relation Acc	Location Acc
HPGM [5]	0.4701	0.5424	0.6464	0.4812	0.5179	-	-
Graph2Plan [11]	0.6796	0.7402	0.8683	0.4476	0.7575	0.9358	0.9805
FP-FGNN	**0.8685**	**0.9165**	**0.8901**	**0.5954**	**0.8019**	**0.9680**	**0.9913**

Table 2. Box-level,Pixel-level and Constraints-level comparisons. We train and evaluate all three models on the same dataset.

the pixel-wise cross-entropy loss with room categories as class variables. Note that out model does not rely on additional losses, such as interior-loss, coverage-loss or mutex-loss *etc*., as used in Graph2Plan [11].

5. Experiments

5.1. Experiments Setup

Dataset. We conduct experiments on the large-scale RPLAN dataset [33], which consists of more than 80729 vectorized floorplan layouts. Since our model is based on graph-constrained layout design, we use its updated version released by Graph2Plan [11]. This version contains a corresponding room-adjacency graph for each floorplan. Overall, there are 15 room-types including *'LivingRoom, MasterRoom, Kitchen, Bathroom, DiningRoom, ChildRoom, StudyRoom, SecondRoom, GuestRoom, Balcony, Entrance, Storage, Wall-in, External, ExteriorWall'* and the room-adjacency types are shown in Tab 1. Each room node comes with attributes of position and size. The method of generating room positions and size is as described in [11] and [5]. Specifically, the floorplan is divided into even-sized grids and the position of the room is the grid where its centroid belongs to. The size of rooms is represented by the normalized area of the room's bounding box. For validation, we use the splits released by [11], which contains 56511 samples for training and 12108 for validation/testing.

Evaluation. Unlike other floorplan design methods that build generative models and evaluate based on Diversity and Compatibility [18, 18, 20, 21, 31], our method focuses on accurately predicting room bounding box coordinates. We primarily use *Intersection Over Union (IOU)*, both Macro/Micro, for box-level predictions, and pixel IOUs with Accuracy for pixel-level assessment. While our main emphasis is on box-level metrics for the initial stage of bounding box prediction, pixel-level metrics also offer insights into layout quality. Furthermore, we assess the preservation of constraints against ground truth, particularly focusing on the accuracy of Relations and Locations, which is central to our model's motivation.

Comparisons. Our main baseline methods for comparison are Graph2Plan [11] and HPGM [5]. These models run GNN on the adjacency graph to predict bounding boxes before generating the layout. To best of our knowledge, these models are the only published methods that are predictive models and take in a graph as input and output the room bounding boxes along with the floorplan layout. Note that Graph2Plan is made generative with a separate retrieval process. Unlike Graph2Plan, HPGM does not take in in-

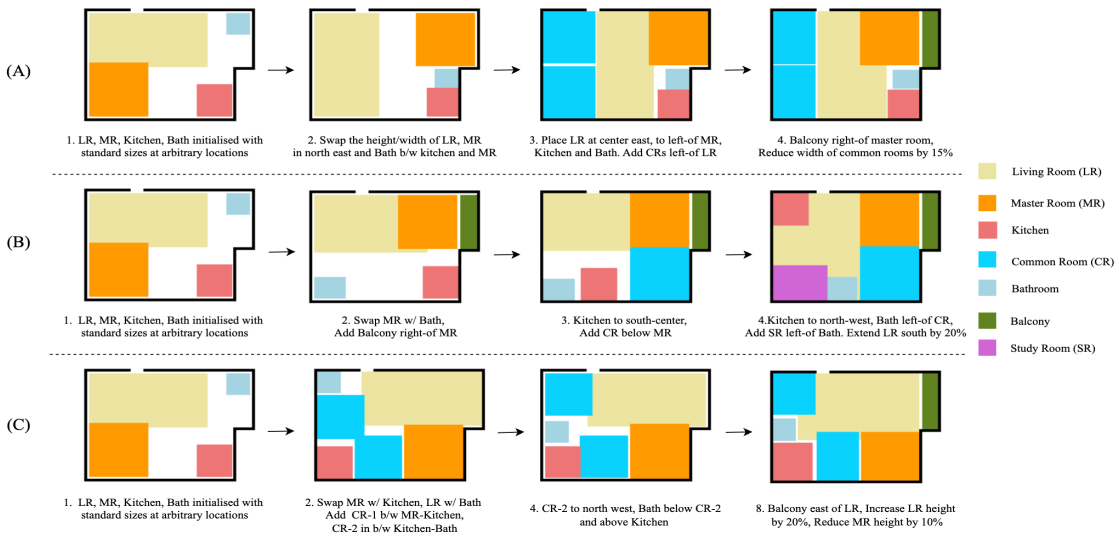

Figure 4. Illustration of iterative design with user iteration. Starting from the same arbitrary partial layout, the interaction process results in three different final layout. The room sizes are automatically adjusted for alignment with boundaries and adjacent rooms.

put boundary. Therefore, for a fair comparison, we include boundary mask into HPGM model the same way as in Graph2Plan. Overall, we make sure to train and test all models with the same input data.

5.2. Results

5.2.1 Comparisons with Baselines

The results, as presented in Tab. 2, demonstrate that our model outperforms the baselines in terms of both box-level and pixel-level metrics, with an impressive 20% improvement observed in box-level IOU metrics. However, we notice that the pixel-level IOU-Macro score is lower for all models, as some classes (*wall-in, Entrance*) have small areas that are hard to be represented with bounding boxes which pull down the macro average since it treats all classes equally. This is supported by class-specific IOU scores provided in Appendix 1.1. Box-level improvements are more pronounced than at pixel-level which further supports our methods effectiveness. The constraints-level scores further validate our model where the location of rooms in predictions matches to ground truth for almost all rooms and preserves relations with high accuracy as well.

5.2.2 Qualitative Results

The benefit of FP-FGNN is equally clear in the visualized qualitative results. We first compare the direct output images of the Graph2Plan and FP-FGNN models with the ground truth images. As shown in Fig. 3, Graph2Plan images visibly contain uneven boundaries for most of the rooms. In contrast, FP-FGNN's output closely resembles the ground truth images with sharper internal boundaries between rooms. We believe the uneven room-boundaries in Graph2Plan is because of the higher overlap in the predicted bounding boxes compared to FP-FGNN, which translates to poor room-boundaries. We show supporting evidence for this by measuring the overlap areas in Appendix 1.2 and additional qualitative results in Appendix 1.4.

5.2.3 Iterative Design with Partial Constraints

In the real world, the layout design process is typically iterative. Users often begin with a vague idea of their desired layout, which then evolves and becomes more refined through an iterative feedback loop. Success in this iterative design process hinges on consistently adhering to the requirements at each step. Our inferential model, characterized by its high fidelity and low inference time, is particularly helpful in such scenarios.

Qualitative analysis Figure 4 illustrates the iterative design process enabled by our model, demonstrating how varying user specifications lead to distinct final layouts. In practical floorplan design, the process often starts with a predetermined boundary. Our model aids users by initially setting up the layout within this boundary, starting with basic elements like the Living Room, Master Room, Kitchen, and Bathroom. These are placed at arbitrary locations in standard sizes (mean size of the room type with an aspect ratio of 1x1) and an aspect ratio of 2x1 for the Living Room. A notable feature of our model is its ability to automatically align walls with the boundary seamlessly, without explicit user input. As users modify or add constraints, our model consistently meets these evolving requirements. This consistent alignment with user inputs enables a productive feedback loop, providing users with the necessary clarity for

further refinement of their requirements. Moreover, with a rapid average inference time of 0.023 seconds, our model is highly efficient for real-time user interaction.

Quantitative analysis We carry out a quantitative evaluation of our model in realistic scenarios, where users have a clear idea of the number and types of rooms but lack a detailed plan for room placement and sizes. This evaluation involves scenarios with different degrees of incomplete information. To emulate this, we randomly select certain number of rooms, and then drop their properties and relational constraints, retaining only the room type information. Note that the dropped rooms are isolated nodes in the graph. In this process, all rooms, except the Living Room, Master Room, and Kitchen, are subject to potential information omission. We train the model with different levels of partial information, and evaluate its effectiveness by measuring its ability to satisfy the constraints: those explicitly defined by the user as well as the complete set of original constraints, including those that were omitted.

	Constraints	Number of Rooms with Omitted Information (K)			
		$K=0$	$K=1$	$K=2$	$K=3$
Ours	Relation(ours)	96.80	96.57/94.00	95.46/87.75	94.69/80.40
	Location(ours)	99.13	99.23/98.26	99.07/95.47	99.05/91.79
G2P	Relation(G2P)	93.58	77.35/74.72	78.79/75.86	78.10/70.44
	Location(G2P)	98.05	94.58/81.04	91.26/76.34	91.80/75.79

Table 3. Accuracy of constraints preserved under partial inputs. We measure Relations and Locations, correctly captured in the output layout, both for specified constraints (first term) and complete set of constraints including omitted ones (second term)

Tab. 3 shows that the preservation of both relation and location constraints for specified rooms is highly effective. It also reveals that the overall accuracy, accounting for dropped constraints, remains stable, despite an increase in the number of dropped rooms. This contrasts sharply with the baseline Graph2Plan model, which exhibits a significant decline in constraint accuracy under similar conditions.

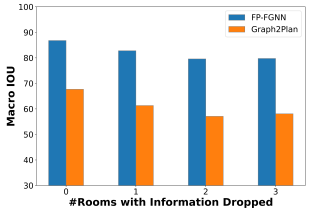

Figure 5. Macro IOU on specified rooms under partial information.

This pattern is also reflected in macro IOU scores (Fig. 5). Notably, our model sustains robust performance, even when information for half of the rooms in the dataset (with an average of 6.79 rooms) is dropped.

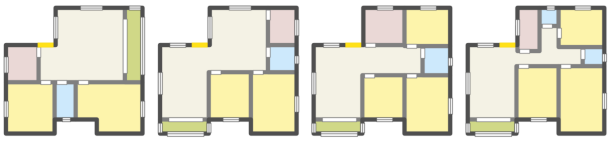

Figure 6. FP-FGNN's diverse generations with same input boundary.

Method	Graph2Plan	RPLAN	Constr-LGen	FP-FGNN
FID	29.26	21.29	21.47	**12.87**

Table 4. FID score comparison of FP-FGNN's generation

5.2.4 Generative Model

In another practical scenario, users find it helpful if they are given with a choice of multiple floorplans to choose from. This use case requires a generative model which can generate diverse floorplans given the same input boundary. Although FP-FGNN is a predictive model, it can be easily used as part of a generative pipeline for generating diverse layouts. To demonstrate this, we use Graph2Plan [11] methodology for the generative mechanism. In Graph2Plan approach, given an input boundary, we first retrieve its k nearest neighbour boundaries from the training dataset. Then we extract the layout graphs from those examples and use it along with the given input boundary as input to the predictive FP-FGNN model. Fig. 6 shows the diversity in generations of our model given the same input boundary. Furthermore, we quantitatively measure the generated images with the FID [10] score and compare with baselines Graph2Plan [11], RPLAN [33], Constr-LGen [23]. Table 4 shows that FP-FGNN generation significantly outperforms the baselines.

6. Conclusion

In this paper, we address the problem of generating layouts under spatial constraints, particularly in the context of floorplan design. We propose a novel approach that employs factor graphs to capture the pairwise and higher-order dependencies between the latent variables representing the rooms' bounding boxes and spatial attributes. Our method employs message-passing on the bipartite factor graph, forming a factor graph neural network that effectively leverages the available domain knowledge to produce floorplans that meet the desired specification. Our results demonstrate a significant improvement in IOU scores over existing methods. In future work, we aim to extend FP-FGNN to learn to generate floorplan layouts from language descriptions [16].

Acknowledgements: This research is supported in part by the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (No. AISG2-RP-2020-016).

References

[1] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13642–13652, 2021. 1, 2

[2] Yunning Cao, Ye Ma, Min Zhou, Chuanbin Liu, Hongtao Xie, Tiezheng Ge, and Yuning Jiang. Geometry aligned variational transformer for image-conditioned layout generation. In *In Proc. of ACM Multimedia*, 2022. 2

[3] Kai-Hung Chang, Chin-Yi Cheng, Jieliang Luo, Shingo Murata, Mehdi Nourbakhsh, and Yoshito Tsuji. Building-gan: Graph-conditioned architectural volumetric design generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11956–11965, 2021. 2

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017. 1, 5

[5] Qi Chen, Qi Wu, Rui Tang, Yuhan Wang, Shuai Wang, and Mingkui Tan. Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12625–12634, 2020. 2, 3, 6

[6] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *In Proc. of UIST*, 2017. 2

[7] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647, 2020. 2

[8] Mohammed Haroon Dupty and Wee Sun Lee. Neuralizing efficient higher-order belief propagation. *arXiv preprint arXiv:2010.09283*, 2020. 2, 3

[9] Feixiang He, Yanlong Huang, and He Wang. iplan: interactive and procedural layout planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7793–7802, 2022. 2

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 8

[11] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, 39(4):118–1, 2020. 1, 2, 3, 4, 5, 6, 8

[12] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 88–96, 2021. 2

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[14] Graziella Laignel, Nicolas Pozin, Xavier Geffrier, Loukas Delevaux, Florian Brun, and Bastien Dolla. Floor plan generation through a mixed constraint programming-genetic optimization approach. *Automation in Construction*, 123: 103491, 2021. 1, 2

[15] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. Neural design network: Graphic layout generation with constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 491–506. Springer, 2020. 1, 2

[16] Sicong Leng, Yang Zhou, Mohammed Haroon Dupty, Wee Sun Lee, Sam Joyce, and Wei Lu. Tell2design: A dataset for language-guided floor plan generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14680–14697, 2023. 8

[17] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. Layoutgan: Generating graphic layouts with wireframe discriminators. *arXiv preprint arXiv:1901.06767*, 2019. 1, 2

[18] Ziniu Luo and Weixin Huang. Floorplangan: Vector residential floorplan adversarial generation. *Automation in Construction*, 142:104470, 2022. 6

[19] Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–12. 2010. 2

[20] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 162–177. Springer, 2020. 1, 2, 6

[21] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13632–13641, 2021. 1, 2, 6

[22] Gizem Özerol and SEMRA ARSLAN SELÇUK. Generation of architectural drawings through generative adversarial networks (gans): A case on apartment plan layouts. *Architectural Research Current Studies and Future Trends*, page 35, 2021.

[23] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6690–6700, 2021. 1, 2, 8

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[25] Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 1: Methodology. *Computer-Aided Design*, 45 (5):887–897, 2013. 2

[26] Eugénio Rodrigues, Adélio Rodrigues Gaspar, and Álvaro Gomes. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: Validation and performance tests. *Computer-Aided Design*, 45(5):898–910, 2013. 2

[27] Julian F Rosser, Gavin Smith, and Jeremy G Morley. Data-driven estimation of building interior plans. *International Journal of Geographical Information Science*, 31(8):1652–1674, 2017. 2

[28] Victor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 685–693. PMLR, 2021. 2, 3

[29] Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. *arXiv preprint arXiv:2211.13287*, 2022. 2

[30] Paloma Sodhi, Eric Dexheimer, Mustafa Mukadam, Stuart Anderson, and Michael Kaess. Leo: Learning energy-based models in factor graph optimization. In *Conference on Robot Learning*, pages 234–244. PMLR, 2022. 2

[31] Jiahui Sun, Wenming Wu, Ligang Liu, Wenjie Min, Gaofeng Zhang, and Liping Zheng. Wallplan: synthesizing floorplans by learning to generate wall graphs. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022. 1, 2, 6

[32] Abhinav Upadhyay, Alpana Dubey, Suma Mani Kuriakose, and Shaurya Agarawal. Floorgan: Generative network for automated floor layout generation. In *Proceedings of the 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD)*, pages 140–148, 2023. 2

[33] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 1, 2, 4, 6, 8

[34] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *In Proc. of ICCV*, 2013. 2

[35] KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard Zemel, and Xaq Pitkow. Inference in probabilistic graphical models by graph neural networks. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 868–875. IEEE, 2019. 2

[36] Zhen Zhang, Fan Wu, and Wee Sun Lee. Factor graph neural networks. *Advances in Neural Information Processing Systems*, 33:8577–8587, 2020. 2, 3

[37] Zhen Zhang, Mohammed Haroon Dupty, Fan Wu, Javen Qinfeng Shi, and Wee Sun Lee. Factor graph neural networks. *Journal of Machine Learning Research*, 24(181): 1–54, 2023. 2, 3

[38] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *In Proc. of ICDAR*, 2019. 2