JOURNAL OF ASIAN ARCHITECTURE AND BUILDING ENGINEERING
https://doi.org/10.1080/13467581.2025.2568756

Taylor & Francis
Taylor & Francis Group

ARCHITECTURAL PLANNING AND DESIGN

OPEN ACCESS Check for updates

Hexagonal cellular automata and graph theory for procedural spatial layout generation

Chelsea Shan Xian Ng[a], Chun-Hsien Chen[a] and Peer Mohideen Sathikh[b]

[a]School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore, Singapore; [b]School of Art, Design and Media, Nanyang Technological University, Singapore, Singapore

ABSTRACT

Spatial layout planning involves arranging spaces to meet functional and aesthetic requirements, a process increasingly automated to save time and explore diverse design alternatives. This study aims to automate the spatial layout planning process using a procedural approach chosen for its flexibility, control, and reduced reliance on large, high quality data sets. The proposed methodology introduces three core algorithms based on hexagonal cellular automata and graph theory, with its versatility showcased across mock examples of public spaces and a real-world case study. The generated layouts effectively satisfy the user-defined design objectives of area, adjacency, and design boundary. A post-processing step converts the hexagonal layouts into rectangular forms with orthogonal corners to align better with conventional architectural layouts while adhering to design constraints. This approach provides a potential foundation for developing automated, user-interactive tools, capable of addressing complex design objectives in architectural contexts. Future work includes the potential integration of visibility and circulation metrics through space syntax analysis and agent-based modelling to enhance wayfinding, traffic flow, and user experience of spatial layouts.

ARTICLE HISTORY
Received 7 February 2025
Accepted 25 September 2025

KEYWORDS
Spatial layout planning; computational layout generation; procedural generative design; cellular automata; graph theory

1. Introduction

Spatial layout planning is a fundamental phase in architectural design, involving the arrangement of space to satisfy different constraints and objectives (Lobos and Donath 2010; Okhoya et al. 2022). Designers must strike a balance between the geometrical requirements, such as the area of rooms and their dimensions, with the topological requirements, such as adjacency between the rooms (Calixto and Celani 2015; Jiang, Wang, and Ma 2023; Ko et al. 2023). This process can prove to be inherently complex and highly iterative, requiring significant time spent and relying heavily on the designer's expertise and intuition to create design variations (Aalaei et al. 2023; Kim and Cho 2020; Liao et al. 2021; Nauata et al. 2020). This can hinder the design exploration and planning process as the design variations created are limited and can be inexhaustive. To address these challenges, researchers have explored the use of computational methods and approaches to automate the spatial layout planning process.

Machine learning (ML) techniques have shown promise in recent years to generate diverse spatial layout variations, utilising models such as generative adversarial networks (GANs) and graph neural networks (GNNs) (Chaillou 2020; Hu et al. 2020; J. Li et al. 2019; Liu et al. 2024; Nauata et al. 2020, 2021). These methods capitalise on large, high quality data sets of historical data to learn spatial relationships, and thereby generate layouts that can mimic real-world design principles (Chaillou 2020; Ko et al. 2023; Nauata et al. 2020, 2021). One such work is HouseGAN which employs a graph-constrained GAN trained on the LIFULL data set to generate residential layouts based on a given bubble diagram representing relational and functional requirements (Nauata et al. 2020). Another prominent example is Graph2Plan, which combines a graphical approach with a deep learning framework (Hu et al. 2020). Users are allowed to specify design objectives and constraints which guide the retrieval of layout graphs from the RPLAN data set (Hu et al. 2020). Using convolutional processing on the input of the building boundary and layout graphs, the retrieved layout graphs contain information to be learned on the desired number of rooms, together with their adjacencies to guide the generation process for generating layouts (Hu et al. 2020). However, a significant challenge faced by such ML approaches is the high dependency on the quality and quantity of training data (Aalaei et al. 2023; Bayraktar Sari and Jabi 2024; C.-W. Zhao, Yang, and Li 2021). This limitation is made more apparent as properly annotated data and large data sets are still scarce in the field of architecture,

CONTACT Chun-Hsien Chen mchchen@ntu.edu.sg School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore, 639798, Singapore

© 2025 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group on behalf of the Architectural Institute of Japan, Architectural Institute of Korea and Architectural Society of China.
This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

engineering, and construction (Chaillou 2020; Okhoya et al. 2022; C.-W. Zhao, Yang, and Li 2021). Additionally, ML models are widely perceived as "black boxes" (Bayraktar Sari and Jabi 2024; Chen et al. 2022; Lipton 2018) with little transparency or user control, which can limit their application in creative fields like architecture.

Another class of approaches is procedural modelling, a rule-based method that does not rely on extensive training data sets (Smelik et al. 2014). Procedural modelling leverages algorithms to provide designers with more transparency and agency to influence the generation process (Chen et al. 2022; Kim and Cho 2020; Lipton 2018). This is particularly advantageous in contexts where data is limited, or where the design process can benefit from flexibility and iterative control while maintaining a degree of automation (Kim and Cho 2020). In addition, procedural modelling avoids any biases that are inherent in ML models trained on historical data, hence enabling the exploration of more novel and diverse solutions (Aalaei et al. 2023; C.-W. Zhao, Yang, and Li 2021).

Shape grammars, for instance, are a rule-based procedural approach that generates designs through application of predefined transformation rules (Jowers, Earl, and Stiny 2019; Stiny and Gips 1972; J. Wang, Zhou, and Zhang 2025). Paulino et al. applied shape grammars to redesign Sobrado buildings into affordable housing while accounting for factors such as daylight, circulation, and structural stability (Paulino, Ligler, and Napolitano 2023). Similarly, graph grammar formalism, as demonstrated by Wang and Zhang, applies rule-driven transformations to adjacency graphs, translating initial graph representations into spatial layouts (X.-Y. Wang and Zhang 2020). Another procedural approach is the use of evolutionary algorithms which are particularly well-suited for solving complex problems with multiple objectives. For instance, Bahrehmand et al. developed an interactive layout recommender where users input dimensional and topological constraints (Bahrehmand et al. 2017). The algorithm evaluates layouts based on spatial qualities like circulation, privacy, and compactness iteratively improving designs over successive generations (Bahrehmand et al. 2017).

Procedural generation is also widely implemented in virtual environments, particularly in game design (Amato 2017; Cogo et al. 2023; Smelik et al. 2014), showcasing its ability for data amplification. From relatively small sets of input data, procedural algorithms dictate how game elements are created and placed, creating a variety of unique configurations (Amato 2017; Smelik et al. 2014). A notable example is Minecraft (Persson and Bergensten 2011), where infinitely large 3D virtual worlds are procedurally generated based on a set of rules governing the spawning of terrain features, biomes, structures, and mobs. Players can use the same seeds to generate identical worlds, making procedural generation reproducible while maintaining its ability to generate varied virtual environments. Rogue-like games further highlight the potential of procedural generation by creating randomized dungeons and levels for each playthrough, offering players endless replay-ability and ensuring each experience feels fresh and engaging.

In view of the advantages of procedural methods, this study introduces a procedural model for spatial layout planning, with additional integration of concepts in cellular automata and graph theory. Cellular automata (CA) are chosen for their ability to dynamically adapt to user-defined design requirements to generate diverse layout options. Apart from being able to function without the need for extensive and well-annotated data sets, CA offers greater flexibility by allowing emergent patterns to form organically through iterative local interactions (von Neumann 1966). Additionally, integrating CA with graph theory enhances the generation process as graphs effectively represent the topological relationships between spaces (Hu et al. 2020; Lobos and Donath 2010; Nauata et al. 2020; Shekhawat et al. 2023; X.-Y. Wang and Zhang 2020), ensuring that adjacency constraints are accounted for. The stepwise generation process when combining CA and graph theory can also potentially allow users to intervene and refine layouts in real time, providing both adaptability and interpretability. The proposed approach thus aims to satisfy user-defined geometrical and topological constraints while maintaining the user's control over the design process. In addition to the core algorithms, here we also introduce a post-processing step to the generated layouts to create more regular and orthogonal room shapes in order to further enhance their practicality for real-world architectural applications.

2. Spatial layout generation with hexagonal cellular automata and graph theory

Cellular automata (CA) are powerful computational models that are used to simulate complex systems (von Neumann 1966). The foundational idea behind CA was first introduced by John von Neumann while developing a model for self-replicating systems in biology (Batty, Couclelis, and Eichen 1997; von Neumann 1966). The model used mathematical properties of cellular structures and growth processes to demonstrate how complex, organized structures as seen in biological organisms could emerge from simple local interactions between individual units, such as cells (Ulam 1952). A typical cellular automaton is executed through the application of simple, local rules to cells in a grid (von Neumann 1966). Each cell is assigned a finite state which evolves based on rules that depend on its current state and the state of neighbouring cells

over iterations (von Neumann 1966). These local interactions among cells thus gives rise to global patterns and emergent behaviours, a characteristic that has been leveraged across various domains from simulations of natural phenomena to game design. CA has been applied to model spread of diseases (Sirakoulis, Karafyllidis, and Thanailakis 2000), fire propagation (Hernández Encinas et al. 2007), and crowd dynamics (Li et al. 2019; Zhao et al. 2021; Zhong, Zhai, and Chen 2023).

In game design, CA has been used to generate dynamic virtual environments and procedural content (van der Linden, Lopes, and Bidarra 2014). CA-based systems can simulate physical interactions and environmental dynamics in games like *Noita*, (Purho et al. 2020) or generate procedural terrain and dungeon structures like in *Terraria* (Omer, Spinks, and Bedna 2011), offering players a dynamic and ever-changing virtual world to explore. These applications of CA showcase its ability to create complex designs from simple rules, and thus make it an ideal approach to generate context-sensitive layouts to tackle spatial design problems. The challenge lies in satisfying both geometrical and topological constraints, that is, attaining target room sizes and fulfilling planned adjacencies, respectively. CA can be used as a model of how rooms are allowed to grow to reach their target size, and also interact with surroundings and neighbouring spaces. For instance, the cells in a grid can represent part of a room or space, and its state can be used to represent whether the cell belongs to a specific room or not. The update rules will be used to dictate how each room expands based on specified adjacencies, and boundary constraints. Iterating the update rules over time evolves the layout while satisfying requirements and maintaining properties of geometry and topology.

A 2D design space can be discretised into a regular grid of cells which will serve as the basic units of the CA system. The choice of the grid type is crucial in determining how the CA behaves and how efficiently it can model spatial relationships. There are three regular polygons that can tessellate a 2D plane without gaps or overlaps: an equilateral triangle, a square, or a regular hexagon. Among these, the most common grid type is that of a square grid. In a square grid, a given cell has four directly adjacent cells (Figure 1ai). These four cells are known as the von Neumann neighbourhood of the given, centre cell (Nugraha et al. 2021). In addition to these, there are four diagonal cells that share only a corner with the centre cell (Figure 1aii). These corner cells, together with the von Neumann neighbourhood, would form what is known as the Moore neighbourhood (Figure 1aii) (Nugraha et al. 2021). However, as the corner neighbouring cells share only a single point with the centre cell, their adjacencies are ambiguous. For instance, two rooms that are diagonally adjacent in a square grid system may not be perceived as directly adjacent in reality. This can potentially lead to difficulties when defining adjacency in the layout generation process.

The second polygon to tessellate a 2D plane is the triangle. By conducting a similar analysis, however, of the total twelve neighbouring cells, only three are directly adjacent to the centre cell (Figure 1aiii), while nine of them share a corner with the centre cell (Figure 1aiv). This can imply that the triangular grid system may be less effective for applications where maximising direct adjacencies is crucial. In contrast, a hexagonal grid system has the greatest number of directly adjacent neighbouring cells, and no cells that share a corner with the centre cell (Figure 1av). This removes ambiguity of defining adjacencies between the cells as all neighbours are directly adjacent to the centre cell. Maximizing the number of direct neighbours with a hexagonal grid system would also allow for a more efficient implementation of adjacency in the spatial allocation process. Using a hexagonal grid system also minimises diagonal bias, which is an issue with square grids where diagonal paths are longer than horizontal and vertical paths, creating unequal distances from a centre cell to its directly adjacent and corner cells. A hexagonal grid system ensures equal distance between all adjacent cells, resulting in isotropic movement across the grid. Therefore, in this work, a hexagonal grid system is used as the basis for developing and executing CA for spatial layout planning.

In this work, we look towards graph theory to represent the relationships between spaces in the spatial layout planning process (Keleş, Takva, and Çakıcı 2025). Traditional bubble diagrams, which are used to visually capture area and adjacency information for spatial layout planning, are graphs in essence. Each space is represented by a vertex, and the edges between the vertices represent the adjacencies between the spaces (Hu et al. 2020; Lobos and Donath 2010; Nauata et al. 2020; Shekhawat et al. 2023; X.-Y. Wang and Zhang 2020). These relationships can be formalised to develop algorithms that generate spatial layouts that adhere to adjacency requirements. In the spatial layout planner developed here, graph theory is used to determine the order of generating spaces within the layout. The concept of rooting trees is extended to adjacency graphs, such that spaces can be sorted into different levels and classified according to their respective adjacencies.

A tree is a type of graph that does not contain any cycles (Figure 1bi) (Valiente 2021). This implies that there is exactly one path between any two vertices (Valiente 2021). By designating one vertex as a root,

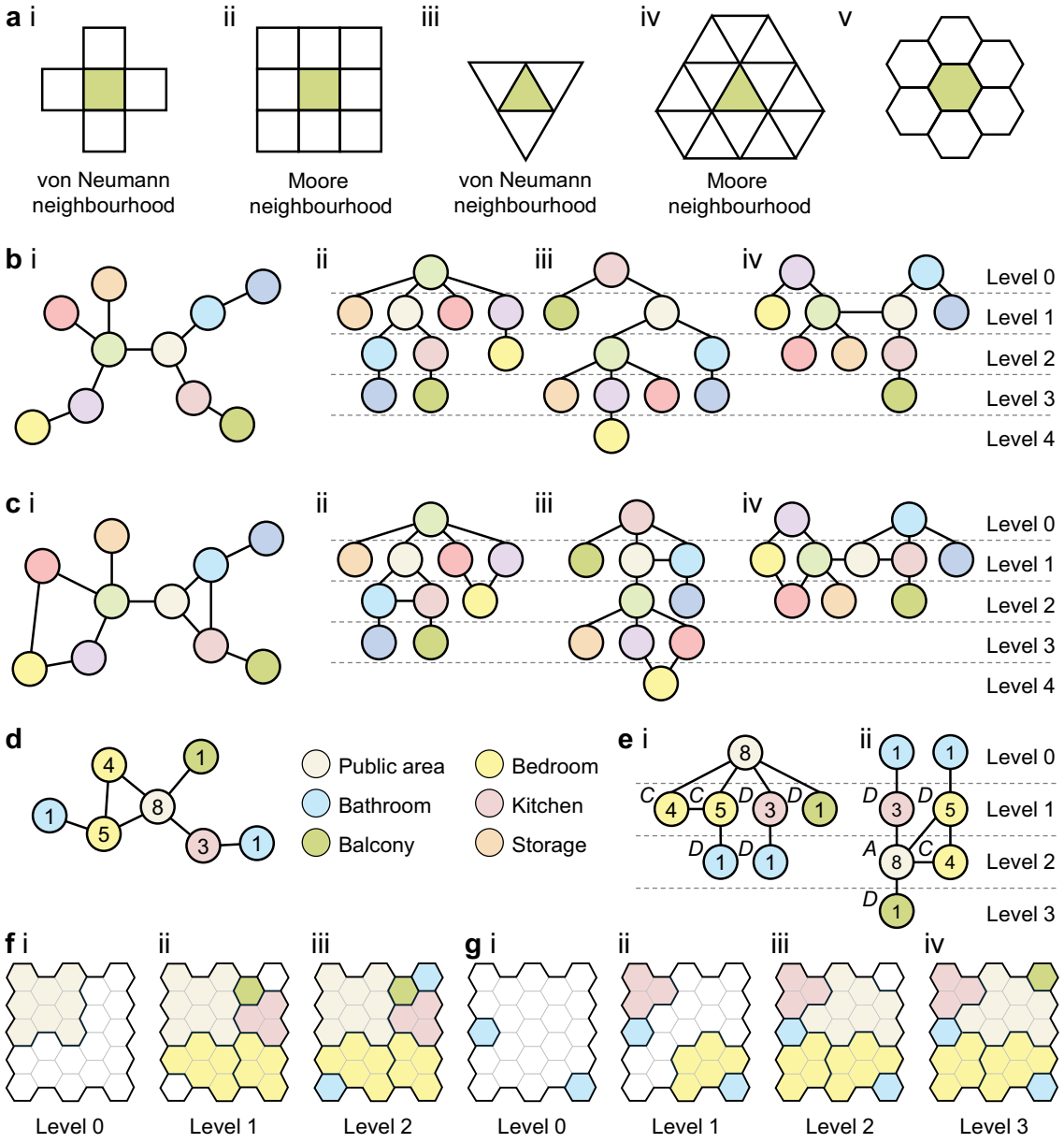

Figure 1. Cellular automata and graph theory used for spatial layout generation. a) Neighbourhoods of a square, triangular, and hexagonal grid system. b) Organisation into levels for trees and c) Graphs including cycles. d) Illustration of an example residential layout d) Organized into levels with f-g) layouts generated in a hexagonal grid system.

a tree is converted into a rooted tree with all remaining vertices split into different levels according to their adjacencies and distance from the selected root vertex (Figure 1bii, iii) (Valiente 2021). The vertices that are connected to and below a particular parent vertex are known as its child nodes (Valiente 2021). Except for the designated root node which has no parents, each child node in a tree has exactly one parent (Figure 1bii, iii) (Valiente 2021). However, in some cases, a tree adjacency graph may not be sufficient to represent more complex spatial layouts as sometimes, multiple spaces may need to be selected as starting points or anchors for the rest of the layout. For instance, the lobby, conference room, and main office can all serve as central key points around which the remaining spaces of an office building are organised. Apart from having to select multiple roots, layouts may often be designed to have cycles within its adjacency graph. Taking a residential layout, for example, the kitchen may be adjacent to the dining room, which is in turn adjacent to the living room. If the living room is also adjacent to the kitchen, it would form a cycle in the adjacency graph. While the idea of rooting trees does not conventionally include designating multiple roots nor rooting graphs with cycles (Valiente 2021), here we propose to do so to encompass the generation of more complex spatial layouts.

A tree can still be organised into levels with multiple roots selected according to their adjacencies and distances from the selected root vertices (Figure 1biv). However, the edges or adjacencies no longer only exist to connect vertices between different levels, but also to connect vertices within the same level (Figure 1biv). With cycles introduced into an adjacency

graph (Figure 1ci), it can still be rooted in a similar way to a tree by also organising vertices into levels based on their adjacencies and distance from selected root(s) (Figure 1cii–iv). Both acts of designating multiple roots and introducing cycles in the adjacency graph mean that child nodes can now have multiple parents instead of having exactly one parent, as with single rooted trees (Figure 1cii–iv). Based on these modifications, a general classification can be created for the non-root nodes of adjacency graphs with singular or multiple roots. In addition to being classified into different levels, each non-root node can be generally classified into four types:

Type A: Has multiple parents and is adjacent to other nodes in the same level.

Type B: Has multiple parents and is not adjacent to other nodes in the same level.

Type C: Has a single parent and is adjacent to other nodes in the same level.

Type D: Has a single parent and is not adjacent to other nodes in the same level.

Using this method of organisation and classification when rooting adjacency graphs, with spaces being vertices and adjacencies being edges, the order of generating and completing spaces in a layout follows the levels of the graph. When designers want to fix the positions of certain spaces, it is likened to designating the roots within the adjacency graph. These user-defined positions and spaces in the layout will be assigned cells and completed first. A space is considered to be completed when it has attained its target area or number of cells, and also when it has established its target adjacencies with other already created spaces. To fulfil the area requirement, rules based on hexagonal CA are used to allocate cells to the space. After the spaces sorted into level 0 (root spaces) are created and completed, subsequent spaces are created level by level. In each level, the starting point or starting cell is selected for each space in such a way that would meet the adjacency requirements of that space with other created spaces. Cells are then added to these starting cells to reach the target area of the space. To illustrate, we use an example of a residential layout with its adjacency graph shown in Figure 1d. Figure 1e shows how this adjacency graph can be converted into rooted graphs, and how the generation process of the layout follows the organisation and classification of the rooted graphs. By selecting a single root or multiple roots, the adjacency graph of this residential layout is organised into different levels, and each vertex can be classified with its respective type (labelled in italics). For simplicity of illustration, the target area of the space is not represented by the size of the vertex or bubble, but instead labelled within the vertex (Figure 1d, e). The first spaces to be positioned in the designed space are the user-selected root spaces (Figure 1fi, gi), with subsequent spaces added on level by level according to the organised adjacency graph (Figure 1fii, iii, gii-iv).

For the purpose of this illustration, the design space boundary is that of a simple rectangular shape consisting of 23 cells, and the positions and shapes of the root spaces and subsequent spaces are arbitrarily decided such that the area and adjacency requirements are satisfied (Figure 1f,g). Next, we describe the CA-based algorithms developed in this work that guide the cell selection process. The procedure includes identifying the starting cells and adding cells to the starting cells, all while maintaining area and adjacency requirements to achieve the desired spatial configuration.

3. Algorithms for spatial layout generation

The procedural model is developed in *Houdini*, a procedural modelling software. Geometry nodes with custom user interfaces are created on this platform to receive the necessary user inputs, and thereafter execute custom functions written with VEX code. To begin, the user has to specify the geometrical constraints of design boundary, which is automatically discretised into a hexagonal cells and pseudo randomly numbered using a geometry node. Thereafter, the user inputs the geometrical and topological objectives of area and adjacencies of each space. The user also has the option to modify a numerical seed which influences the pseudo-random processes within the procedure. The final user input will be to select the starting cells of the root spaces based on the numbered cells in the discretised area. From these inputs, the procedure operates with the hexagonal grid system and generates layouts using algorithms based on the earlier discussed concepts of CA and graph theory. The desired final output is a floor plan showing allocated spaces and boundaries that adhere to the user-defined geometrical and topological objectives. The overview of the procedure is illustrated in Figure 2, which involves a combination of user inputs and automated algorithmic steps to arrive at the final output.

A mock museum layout is created as an example of user input, with Figure 3 showing its bubble diagram and different design boundaries that are discretised into hexagonal cells. The area to be assigned to each space is labelled within the respective bubbles, and its units are m^2. The boundary of the design space represents the physical limits within which the layout is generated. In *Houdini*, it is drawn by the user as a 2D polygon using a geometry node. After discretisation, the cells can be identified and specified by the user using a pseudo randomly assigned number starting from 0. For simplicity of illustration, each cell is set to represent an actual area of 1 m^2. This therefore implies that the target area labelled in Figure 3a is equal to the number of cells to be allocated to that space. The

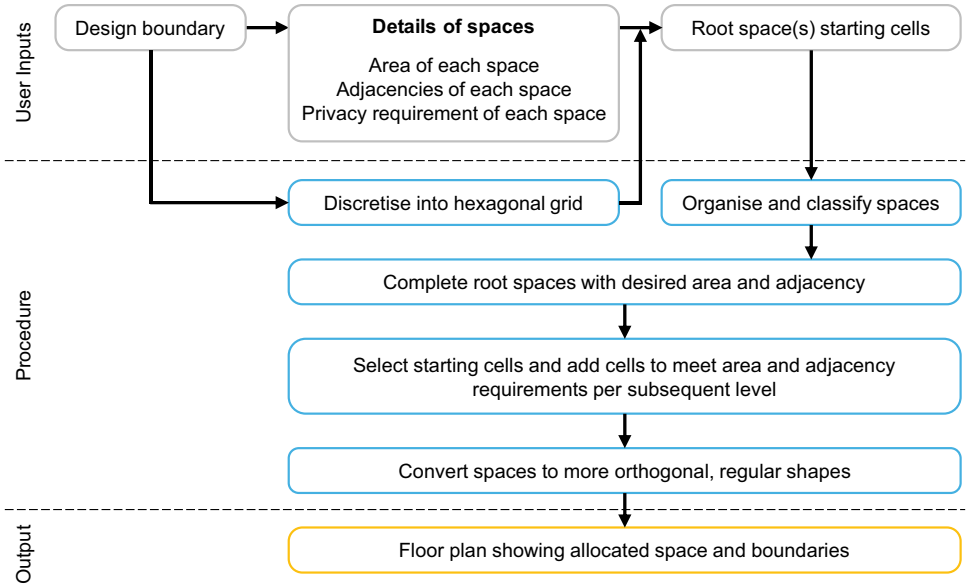

Figure 2. Proposed procedure for layout generation.

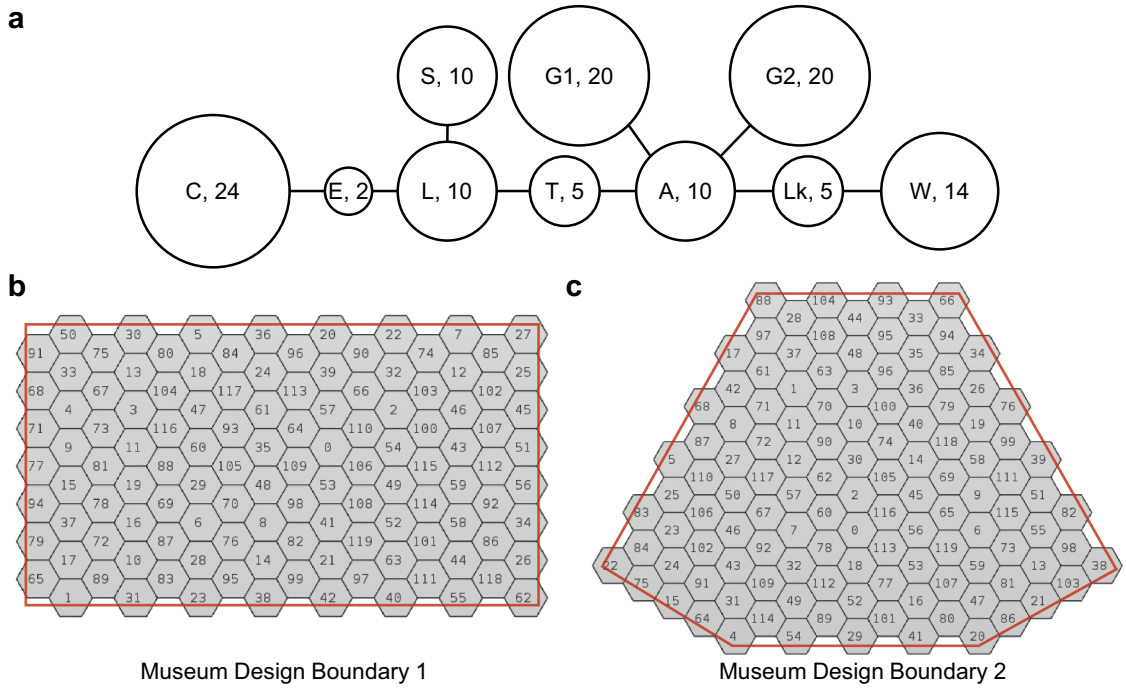

Figure 3. Design inputs for the mock museum. a) Bubble diagram and b-c) design boundaries discretised with the hexagonal grid system.

resolution of the grid, however, can be modified by the user, consequently changing the area which each cell represents. For Figure 3 and subsequent figures representing the mock museum, the names of spaces are abbreviated as follows: A-Atrium, C-Carpark, E-Entrance, G1-Gallery 1, G2-Gallery 2, L-Lobby, Lk-Lockers, S-Shop, T-Ticketing, W-Toilets.

In a separate geometry node, the user specifies the geometrical and topological objectives of the layout through a custom user interface. The user will have to enter the number of spaces to be generated and their corresponding name and area. The user can then enter the total number of adjacencies in the layout and list the pairs of spaces that are to be adjacent. To designate cells to spaces, the user has to reference the discretised design boundary and specify the cell numbers to be assigned to select spaces. Note that at least one space must have at least one cell assigned to it to start. The spaces with user-assigned cells will therefore form the roots of the adjacency graph. Based on the root spaces and adjacencies, the remaining spaces that have yet to be assigned with cells are automatically sorted into their respective levels to determine the generation sequence. Starting with the spaces in level 0 (root spaces), cells are added to each space until their target area is reached. For each subsequent

level, the generation process involves two main steps that are executed recursively: First, a starting cell (or cells) for a space is selected via Algorithm I which identifies neighbouring cells of completed parent spaces and/or created peer spaces. This step ensures that adjacencies of the space are established accordingly. Secondly, Algorithm II adds cells to the selected starting cell(s) until the space meets its area requirements. These two algorithmic steps are executed in each level until all spaces have been generated and completed. Finally, Algorithm III is a supplementary to these first two algorithms, in the event that they are unable to be executed due to there being no available cells to select.

3.1. Algorithm I – cells of the minimum spanning distance

Algorithm I aims to identify the fewest number of cells that connect two or more cell clusters, where a cell cluster in this context can refer to a group of cells or a single cell. We term these as cells of the minimum spanning distance between cell clusters. To achieve this, we start by first examining Dijkstra maps or floor field models using CA (Adamatzky 1996; Khashoggi et al. 2018; Yang, Wang, and Qin 2015; Zhao et al. 2021). Dijkstra maps are inspired by Dijkstra's algorithm whose purpose is to find the shortest path between vertices in a weighted graph (Dijkstra 1959). By systematically updating and evaluating the shortest known distance between the source and each node, the shortest path from the source node can be found iteratively. The principles of Dijkstra's algorithm can be applied to a grid system to find the shortest path between a source cell and all other cells within the grid. To obtain a Dijkstra map, integer values are assigned to each cell, beginning with all immediate neighbouring cells of the cell cluster assigned a value of "1". Subsequent neighbouring cells are assigned values incrementally, increasing the value by 1 as the distance from the original cell cluster grows. Eventually, all cells will have a value that represents their distance from the cell cluster, and this is referred to as the Dijkstra mapping for that cell cluster. From any point in the grid, moving towards the next adjacent cell with the lowest Dijkstra value will trace the shortest path back to the source cell.

The idea behind finding the minimum spanning cells is also similar to finding the minimum spanning tree between vertices of a given graph. The edge weights of the graph can be likened to the minimum number of cells between pairs of spaces or vertices, and the minimum spanning tree would be the minimum number of cells that connects all the spaces. While there are well established algorithms to find the minimum spanning tree in a weighted graph such as Prim's algorithm (Ayegba et al. 2020; Prim 1957) or Kruskal's algorithm (Ayegba et al. 2020; Kruskal 1956), they are not applicable in a grid system as they treat the vertices as points rather than occupying a 2D space. These algorithms also treat edges as discrete rather than having the possibility of overlapping, such as when a cell can be a part of the minimum distance between a pair of vertices (or cell clusters). Therefore, such algorithms will not suffice to find the minimum spanning cells between cell clusters, leading to the development of Algorithm I using Dijkstra mappings for our procedural model.

When using a grid system, there may be multiple shortest paths between two cells or two clusters of cells (as shown in Figure 4a). As such, Algorithm I first identifies the group of candidate cells that contain the possible shortest paths, and thereafter reduces it to obtain one possible shortest path between cell clusters. To identify the group of candidate cells, the Dijkstra values from the respective maps of the involved cell clusters are summed to obtain Mapping 1 (Figure 4b). From Mapping 1, the candidate cells are found from grouping cells starting from the lowest value, such that this group is adjacent to all involved cell clusters (Figure 4b, cells in yellow). If only two cell clusters are involved, the group of candidate cells will typically consist only of the lowest valued cells in Mapping 1 (Figure 4bi). If more than two cell clusters are involved, the group may need to include cells with the next lowest Dijkstra values such that the group is adjacent to all involved cell clusters (Figure 4bii). After obtaining the group of candidate cells in the shortest path, the second step of reducing the group additionally requires Mapping 2, a mapping of the lowest constituent Dijkstra value between the involved cell clusters (Figure 4c).

Among the highest-valued cells in Mapping 1 (cells outlined in blue in Figure 4c), the highest-valued cells in Mapping 2 are compared in a nested loop. In the inner loop, the cells with the highest values from Mapping 2 are removed as long as doing so does not separate the group of candidate cells itself nor does it separate the group from any of the involved cell clusters (Figure 4c). When no more cells can be removed, the cells with the next highest values from Mapping 2 are considered for removal (Figure 4c). This is repeated until the cells valued "1" in Mapping 2 are considered and no more cells can be removed from the group. The outer loop then moves on to execute the inner loop for the cells of next highest values in Mapping 1 (Figure 4cii). This nested loop is essential for reduction as opposed to randomly selecting cells to remove. A counterexample is shown in Figure 4d where randomly removing cells from the group (without separating the group itself nor the group from either of the two cell clusters) can result in erroneous derivation of the shortest path.

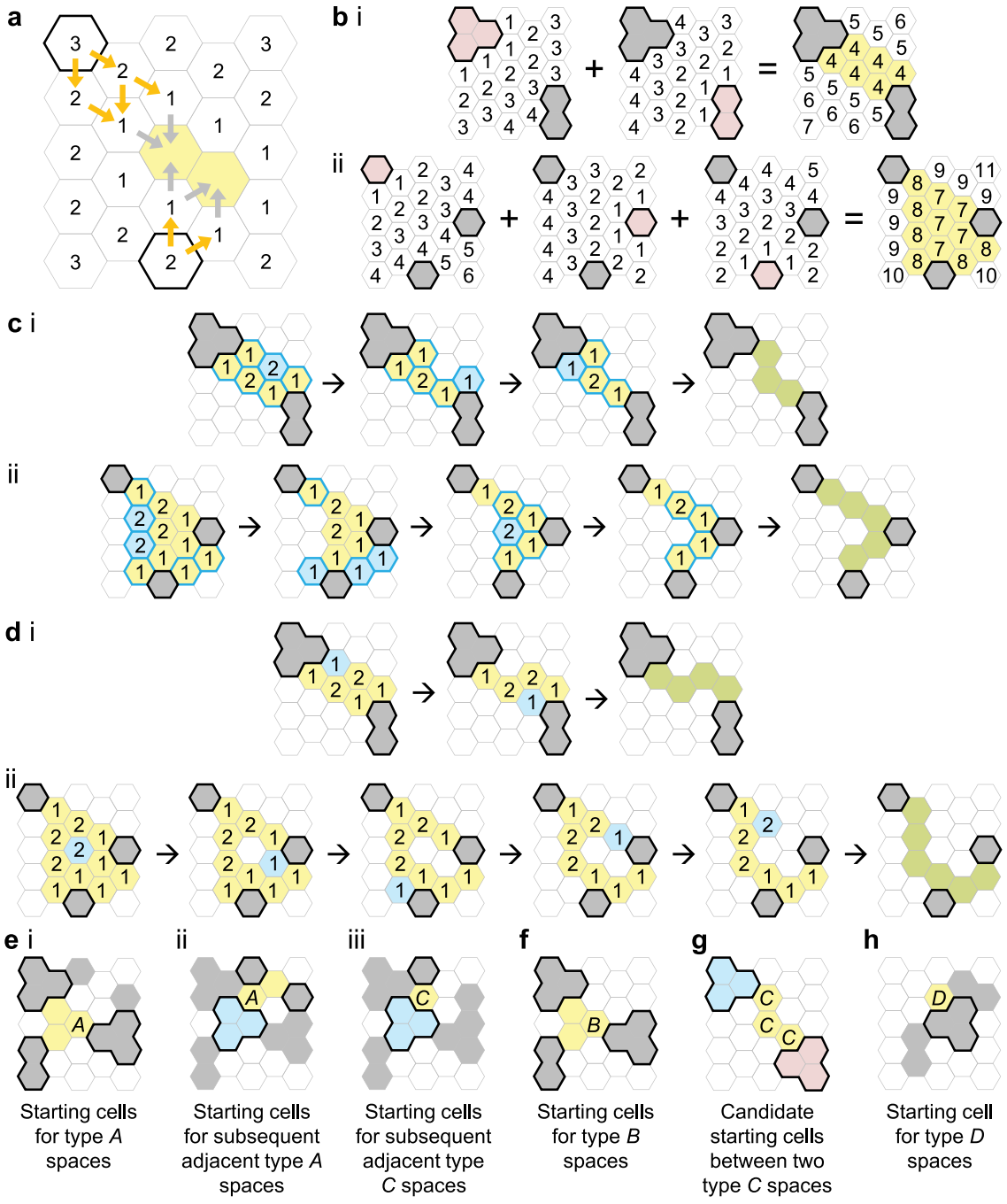

Figure 4. Algorithm I finds the starting cells for establishing adjacencies. a) Multiple shortest paths from any arbitrary cell to a group of cells (in yellow). b) Identifying the group of candidate cells with mapping 1. c) Reducing the group of candidate cells to the cells in the shortest path with mapping 2. d) Erroneous derivation of the cells in the shortest path without using mapping 2 for reduction. e) Use of algorithm I to Find starting cells for type a spaces and spaces adjacent to type a spaces in the same level. f) Finding the starting cells for type B spaces using algorithm I. g) Finding the candidate starting cells between the respective parents for two type C spaces using algorithm I. h) Pseudorandom assignment of starting cell for a type D space.

The starting cells are first selected for spaces with multiple parents, which are both *type A* and *B* spaces using Algorithm I to identify the cells of the minimum spanning distance between all their respective parent cell clusters (parent cell clusters are outlined in black, Figure 4ei, f). This will complete the selection of starting cells for *type B* spaces. Thereafter, the adjacencies within the level are created for *type A* and *C* spaces using cells of minimum distance between the spaces found with Algorithm I as the starting cells of the respective space (Figure 4eii, iii). These adjacencies within the level can be between two *type A* spaces (Figure 4eii), one *type A* and *C* space (Figure 4eiii), or two *type C* spaces (Figure 4g).

For adjacencies with a *type A* space, the cells of the minimum distance are found with their already assigned starting cells (Figure 4eii). For adjacencies with a *type C* space, the cells of minimum distance are found from their parent space cell clusters (Figure 4eiii, g). However, if a *type C* space has

already been assigned starting cells perhaps from creating its other adjacencies to other spaces in the same level, its starting cell cluster will be used instead of the parent space cell cluster to find the minimum distance with. This will complete the selection of starting cells for *type A* and *C* spaces. Finally, the most straightforward selection of starting cells would be for *type D* spaces; only one neighbouring cell of its parent space will be selected pseudo randomly to be the starting cell (Figure 4h). Selecting the starting cells in order of the types of space satisfies the topological requirements of the parent spaces and child spaces between each level, and also that within the levels.

After all starting cells have been selected successfully, Algorithm II is executed next to add cells onto the starting cells of each space in the level to satisfy their geometrical requirements. However, Algorithm I fails when the number of cells in the minimum spanning distance is greater than the number of cells that can be allocated to the spaces based on their target area. The user can be prompted to modify the input in such a case, for instance, to assign more area to the spaces involved or to fix the relative positions to be closer, in order to proceed. Another case where Algorithm I cannot be executed would be if a space has no available, unassigned neighbouring cells. In such cases, the supplementary Algorithm III is executed to rectify this issue.

3.2. Algorithm II – cell addition

Algorithm II concurrently adds neighbouring cells to the starting cells of each space in a level. It is executed iteratively until all spaces in the level have attained their target area, or no more cells can be selected to be added. From the starting cell(s), a space can expand in all directions into available, unassigned neighbouring cells in one iteration. The ability to expand will depend on two factors: Firstly, it depends on the remaining number of cells to be added to reach its target area (n_R). Secondly, it depends on the number of available, unassigned neighbouring cells (n_N). To determine if a cell is assigned to a space in the current iteration, we evaluate bid values in a cell that are put out by spaces to their available, unassigned neighbouring cells. The bid value is defined as the difference between n_R and n_N:

$$Bid\ value = n_R - n_N. \tag{1}$$

All cells are evaluated in one iteration, and a cell is assigned to the space with the highest bid value and if its bid value is ≥0. Figure 5a shows the bid values and addition of cells to the starting cell cluster of a single space. If the bid value of a space is <0, it implies that the space has more available neighbours than remaining number of cells to be added. Hence, if a cell has bid values that are all <0, the final step in the iteration will

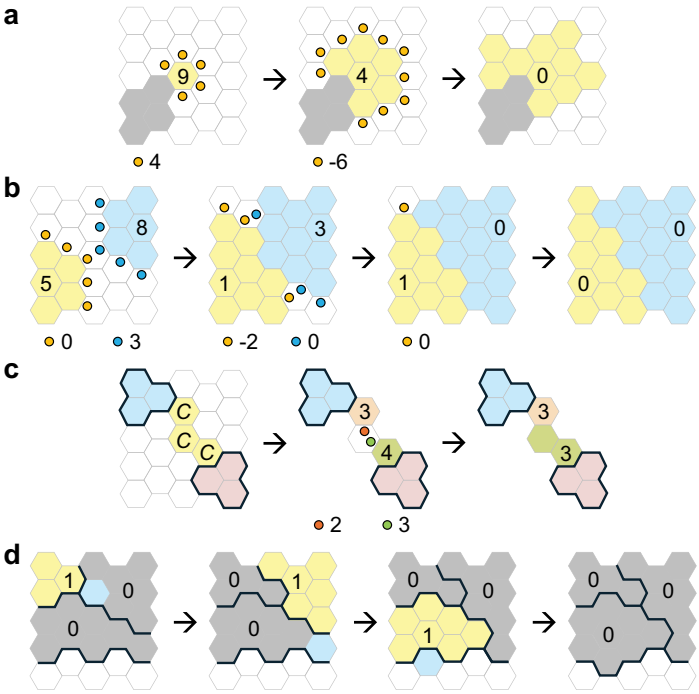

Figure 5. Algorithms II and III add cells to fulfil area requirements. a) Cells are added to a single space using algorithm II based on bid values put out to neighbouring cells. b) Cells added to different spaces simultaneously using algorithm II. c) Algorithm II is also used to assign the minimum spanning cells identified with algorithm I to the respective type C spaces. The bids of the spaces shaded in yellow are indicated by a yellow dot marker, likewise for the spaces in blue, orange, and green. Their respective bid values are indicated below the corresponding figures. d) Selection of already assigned cells using algorithm III results in a displacement effect.

be to pseudo randomly select the available neighbours of the space according to the n_R (Figure 5a).

The usefulness of evaluating a bid value to assign cells is more apparent when there are more than one space in a level (multiple starting cell clusters). Some spaces may share neighbouring cells, implying that in any iteration, one cell can have more than one bid value (Figure 5b). Similarly, in one iteration, a cell is assigned to the space with the highest bid value that is ≥0 in that cell (Figure 5b). This is iterated until all cells have bid values that are <0, implying that $n_R < n_N$ for all spaces in the level. The final step will then be to pseudo randomly select available neighbours of each space based on the respective n_R (Figure 5b). If a cell has been selected in the final step to be added to multiple spaces, it will be assigned to the space with the highest (least negative) bid, and other cells will be selected for the other spaces that were contending for that cell.

Apart from adding cells to selected starting cells, Algorithm II is also used in cases after Algorithm I has identified the minimum spanning cells between *type C* spaces that have not been assigned their starting cells. From the cells of the minimum spanning distance, all other cells are temporarily marked as unavailable (Figure 5c). The cell clusters of Algorithm I will then become the 'starting cells' to which cells will be added using Algorithm II (Figure 5c).

As cells are added to the spaces within the defined design boundary, some spaces may become trapped, i.e., when a bid value is ≥0 but $n_N < 0$., implying that no suitable cells can be found using Algorithm II alone. Hence, Algorithm III is used to allow the selection of unavailable (already assigned) cells.

3.3. Algorithm III – displacement of cells

The selection of starting cells with Algorithm I may fail if a space has no available, unassigned neighbouring cells. Algorithm II may also be unable to find suitable cells to be added to starting cells. Therefore, Algorithm III was developed as a supplement to Algorithms I and II to allow the selection of unavailable (already assigned) cells when necessary. The selection of unavailable cells is done through displacing the cells of neighbouring spaces, while maintaining their already existing required adjacencies as well as the number of cells assigned to them. When a space has $n_R \geq 0$ but $n_N < 0$, one already assigned neighbouring cell is pseudo randomly selected to be reassigned to the space. A cell can only be selected and reassigned if doing so does not result in disconnecting or splitting of any existing spaces, nor does it remove any existing adjacencies (Figure 5d). Note also that cells have been deliberately assigned by the user to a space cannot be considered for reassignment.

After reassignment, the space that the selected cell originally belonged to will have its number of cells decreased by one (Figure 5d, the remaining number of cells required for the space is labelled within the space). Therefore, the reassignment process is repeated for that space to replace the missing cell and maintain its area. The reassignment continues until an unassigned neighbouring cell can be selected instead of an already assigned neighbouring cell. The overall result is that cells can be reassigned added to trapped spaces by displacing the cells of neighbouring spaces while maintaining the areas and adjacencies of existing, completed spaces (Figure 5d).

To summarize, when creating spaces within a single level, Algorithm I is executed first to select the starting cells and satisfy adjacencies. When all spaces within the level have had their starting cells assigned, Algorithm II adds cells to the starting cells to satisfy area requirements of the spaces. Algorithm III is executed only when there are no available neighbouring cells to select for Algorithms I and II, allowing the selection of already assigned cells while maintaining the adjacencies and areas of spaces in all levels. When all spaces within the level have been completed, the generation proceeds to the next level, repeating until all spaces have been completed.

4. Results and discussion

Based on the cells assigned by the user, the root spaces are decided, transforming the respective adjacency graphs into rooted trees, with the remaining spaces sorted into their different levels, indicating the order of creating these spaces after the root space has been generated (Figure 6a). The generated layouts are considered valid as long as they have met the defined objectives of area and adjacency, as well as fit within the user-specified design boundary.

By having the starting cell for the lobby specified, the adjacency graph is rooted as shown in Figure 6ai, two distinct layouts can be generated (Figure 6b, c) for each of the design boundaries shown in Figure 3b-c. The assigned cell number for the lobby can be changed to indicate a different starting position (Figure 6d, using also the boundary in Figure 3b). In addition, more than one cell can be selected for the root space (Figure 6e), which may be desirable for dictating the shape of selected spaces. Besides the lobby, cells can be selected for other spaces to designate them as the new root of the adjacency graph (Figure 6aii). For instance, to anticipate the constraint of sewage and related piping on the location of the toilets, it is selected as a root space by assigning a starting cell to it, creating the rooted graph in Figure 6aii and its resultant generated layout (Figure 6f). Lastly, should the position of more than one space be fixed, it is also possible to select the starting cells for more than one

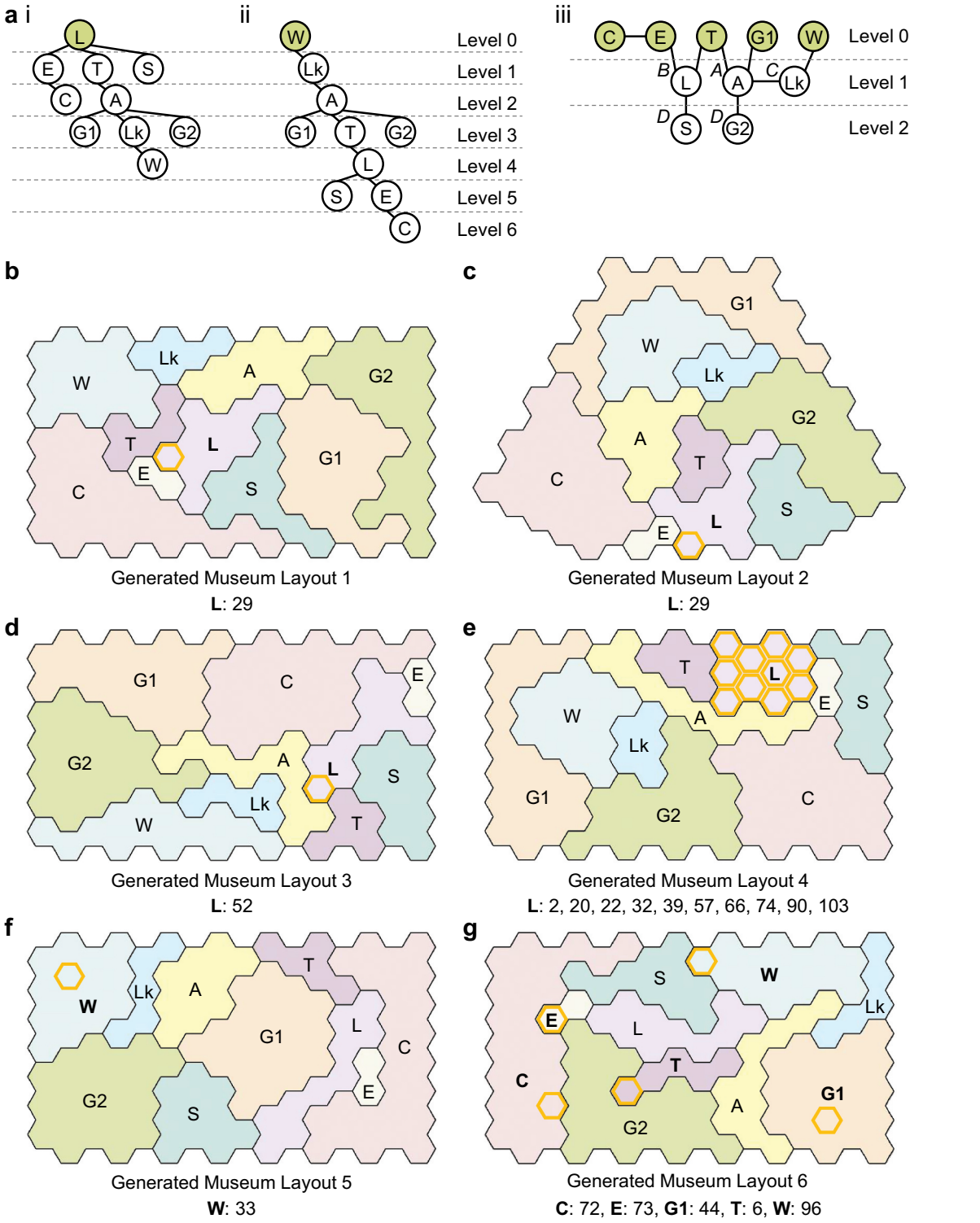

Figure 6. Generated layouts of mock museum. The non-root spaces are annotated with their type classification. The starting cell numbers are labelled below the generated layouts, and their locations are outlined in dark yellow.

space, transforming the adjacency graph into a multi-rooted tree (Figure 6aiii). The corresponding generated layout is shown in Figure 6g.

Apart from the museum, mock layouts were also generated for other types of spaces such as a section of a medical centre, and also a public library, with completely different user inputs shown in Figure 7ai, bi, respectively. Each plan also has a different design boundary, and their discretisation is shown in Figure 7aii, bii. From their respective inputs, their adjacency graphs are organised into levels to dictate the order of generating spaces (Figure 7aiii, biii), successfully outputting their final layouts (Figure 7aiv, biv). For the figures representing the mock medical centre, the names of spaces are abbreviated as follows: C1-Clinic 1, C2-Clinic 2,

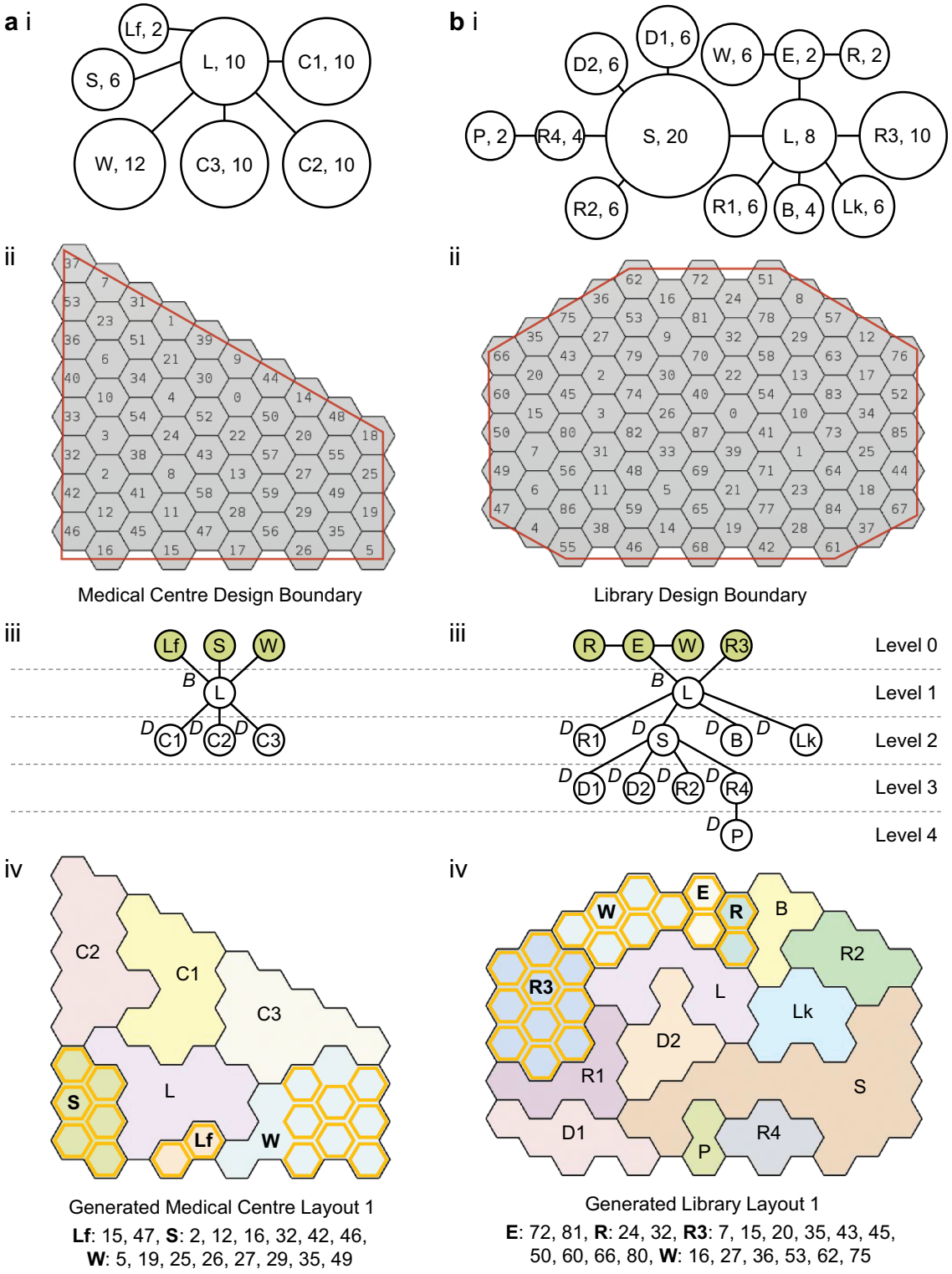

Figure 7. Design input and generated layouts for the mock a) medical centre and b) public library. The starting cell numbers are labelled below the generated layouts, and their locations are outlined in dark yellow.

C3-Clinic 3, L-Lobby, Lf-Lifts, S-Stairs, W-Toilets. For figures representing the mock public library, the names of spaces are abbreviated as follows: B-Borrowing Kiosks, D1-Desks 1, D2-Desks 2, E-Entrance, L-Lobby, Lk-Lockers, P-Printing Room, R-Reception, R1-Seminar Room, R2-Study Room, R3-Multifunction Room, R4-Media Room, S-Main Shelves, W-Toilets.

The generated layouts shown thus far are created from adjacency graphs that do not have cycles. In the proposed approach, the main difference between planning for a layout without cycles and one without is the classification of spaces. For single rooted trees, all non-root spaces are *type D* spaces, whereas multi-rooted trees will include the other types of spaces. We also show that layouts can be generated for layouts with adjacency graphs

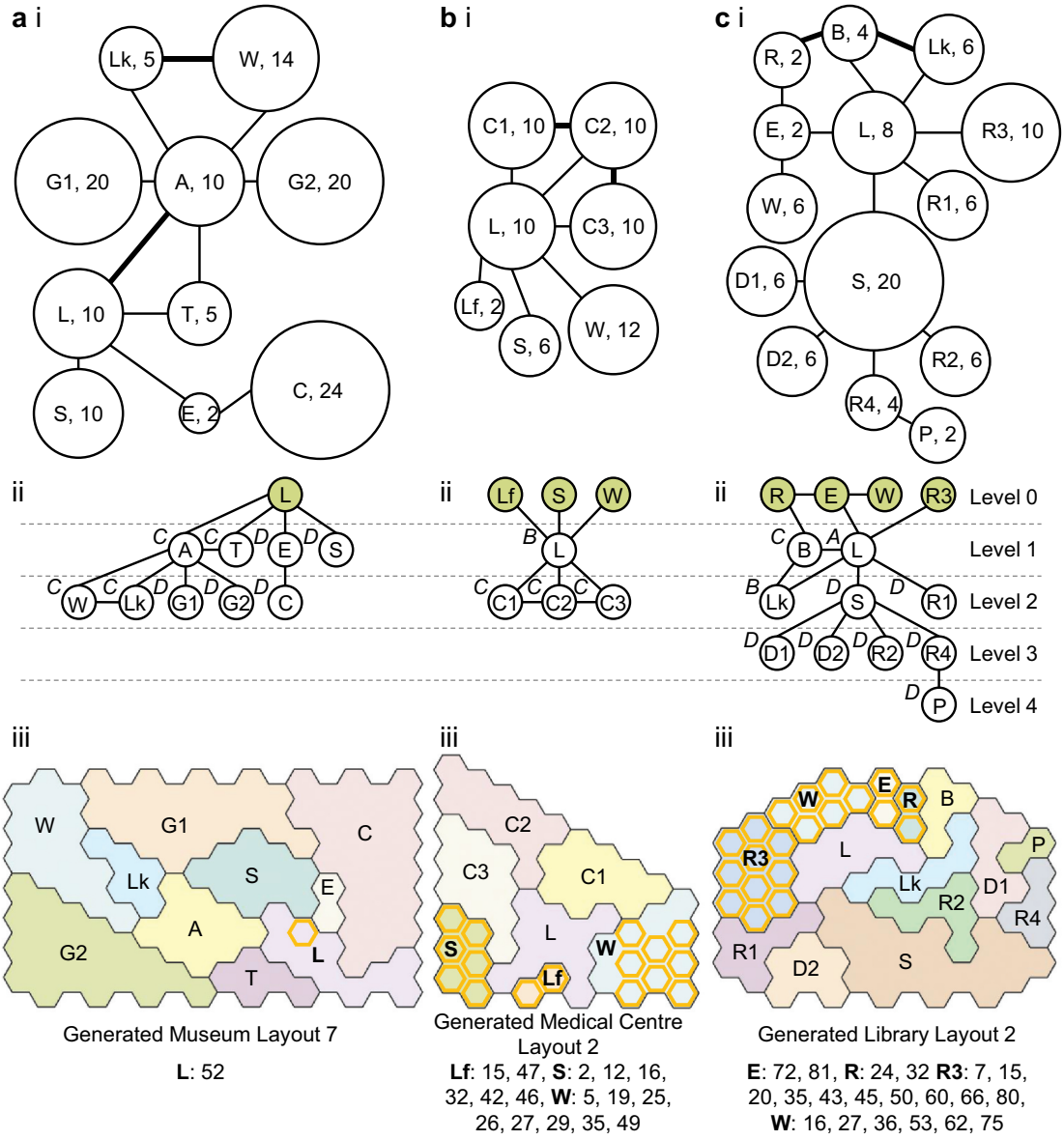

Figure 8. Modified design inputs inclusive of cycles for the mock a) museum, b) medical centre, and c) public library. The starting cell numbers are labelled below the generated layouts, and their locations are outlined in dark yellow.

containing cycles by modifying the three example layouts to include cycles (Figure 8ai, bi, ci for the mock museum, medical centre, and library, respectively). The additional adjacencies are indicated as bold solid lines (Figure 8ai, bi, ci). The spaces are organised into levels based on their respective cells selected for different spaces (Figure 8aii, bii, cii), which leads to their generated layouts (Figure 8aiii, biii, ciii).

4.1. Post-processing

As the floor plans are generated from hexagonal cells to facilitate the attainment of adjacencies, the spaces also tend to adopt hexagonal borders and shapes which may not be commonly seen in conventional architectural design. In real-world scenarios, rectangular room shapes with orthogonal corners remain standard, probably due to ease of construction and practicality for placement of furniture. Therefore, we have created a post-processing step to convert the hexagonal-based layouts into more rectangular and orthogonal room shapes, while preserving the area and adjacencies of the spaces.

The process begins by overlaying the hexagonal grid with a rectangular grid, where the corner points of the rectangular cells correspond to the midpoints of each of the four diagonal sides of a hexagonal cell. As such, the area contained within a rectangular cell can either be fully occupied by part of a hexagonal cell, or shared by parts of up to four different hexagonal cells positioned on its left, right, top, and bottom (Figure 9a). A rectangular cell whose area is fully occupied by part of a hexagonal cell will be assigned to the same space as that hexagonal cell. On the other hand, the space assigned to rectangular cells with area shared between different hexagonal cells will depend on the configuration of the hexagonal cells.

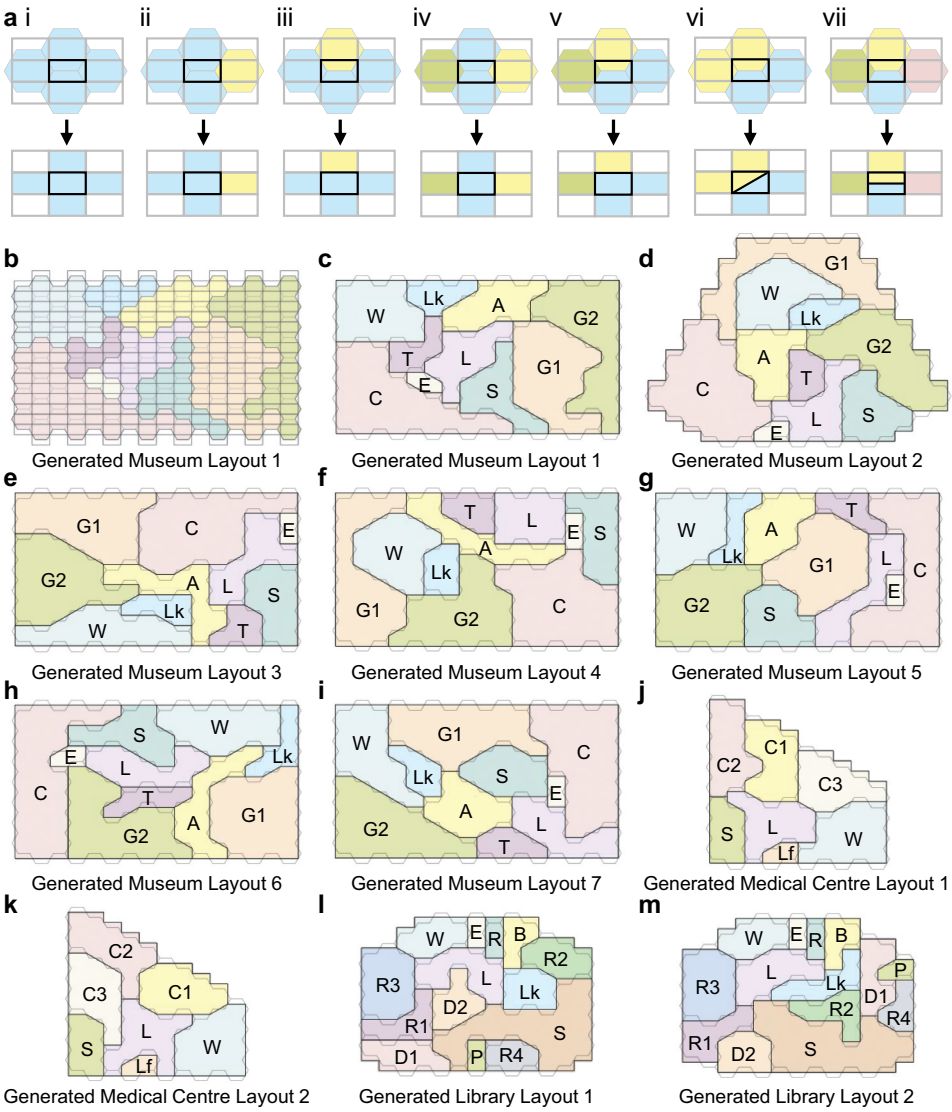

Figure 9. Post-processing of generated layouts to create straight borders between spaces and create more regular room shapes and orthogonal corners.

Referring to Figure 9ai-iii, if three or more of the hexagonal cells sharing the area of the rectangular cell belong to the same space, the rectangular cell is assigned to that space. If two of the hexagonal cells belong to the same space, the rectangular cell is also assigned to that space (Figure 9aiv, v). However, if the two hexagonal cells at the top and side belong to the same space, and the two hexagonal cells at the bottom and the opposite side both belong to another space, the area of the rectangular cell is split diagonally and assigned to the respective spaces (Figure 9avi). Where all cells belong to different spaces, the rectangular cell is split horizontally between the space assigned to the top hexagonal cell and the bottom hexagonal cell, respectively (Figure 9avii). If the top hexagonal cell is not assigned to any space, the rectangular cell will be fully assigned to the space of the bottom hexagonal cell, and the converse also applies. This will also be true if the hexagonal cells on the left and right belong to the same space. Following these transformation rules, the rectangular cells are divided accordingly and assigned to their respective spaces, resulting in straighter borders between spaces and orthogonal corners. The results of the transformation are shown for all the generated layouts that have been presented (Figure 9c-m). It is important to note that the converted layouts still maintain their desired adjacencies while maintaining their relative areas.

5. Discussion

Through leveraging concepts from cellular automata (CA) and graph theory, our proposed methodology is a systematic approach to spatial layout planning. The generated results demonstrate the flexibility of the proposed procedural model in generating spatial layouts that satisfy the user-defined area and adjacency requirements. While the objectives of meeting area and adjacency requirements are foundational in spatial layout planning, there are other important objectives such as circulation and visibility considerations that are not yet explicitly addressed here.

In addition, the model can be further validated for application in real-world scenarios as the presented results are only based on mock layouts. To demonstrate the feasibility of the model in real-world scenarios, we applied it to the Bukit Panjang Public Library (BPPL), located in Bukit Panjang Plaza, a local shopping mall in Singapore. The library underwent a major renovation in 2017, expanding its floor space by 85% to accommodate a growing population (Ng 2017). Its renovated real-world layout has a complex spatial configuration that is ideal for demonstrating the adaptability and potential of the proposed procedural model in redesigning functional public spaces. Using the existing real-world layout as a reference (Figure 10a) (National Library Board, Singapore 2025), the design boundary (Figure 10b), area, and adjacencies have been adapted as user inputs to the model (Figure 10c). The names of the spaces in BPPL are abbreviated as follows: AV-Audio visual section, D-Desks, E-Entrance, E1-Exit 1, E2-Exit 2, E3-Exit 3, L-Lobby, M-Multimedia, P-Program room, S-Seats, S1-Shelf 1, S2-Shelf 2, S3-Shelf 3, S4-Shelf 4, S5-Magazine shelf.

As the library's location is constrained to be within a shopping mall and has to comply with safety regulations and egress guidelines, the locations of the entrance and exits are fixed (to be the same as that in the actual layout of the library). This results in the multi-rooted adjacency graph shown in Figure 10d. Additionally, area for non-public spaces were included into the design boundary (grey areas in Figure 10a), resulting in a larger available area than the total planned space. The generated layout will therefore have redundant cells which can be used for private

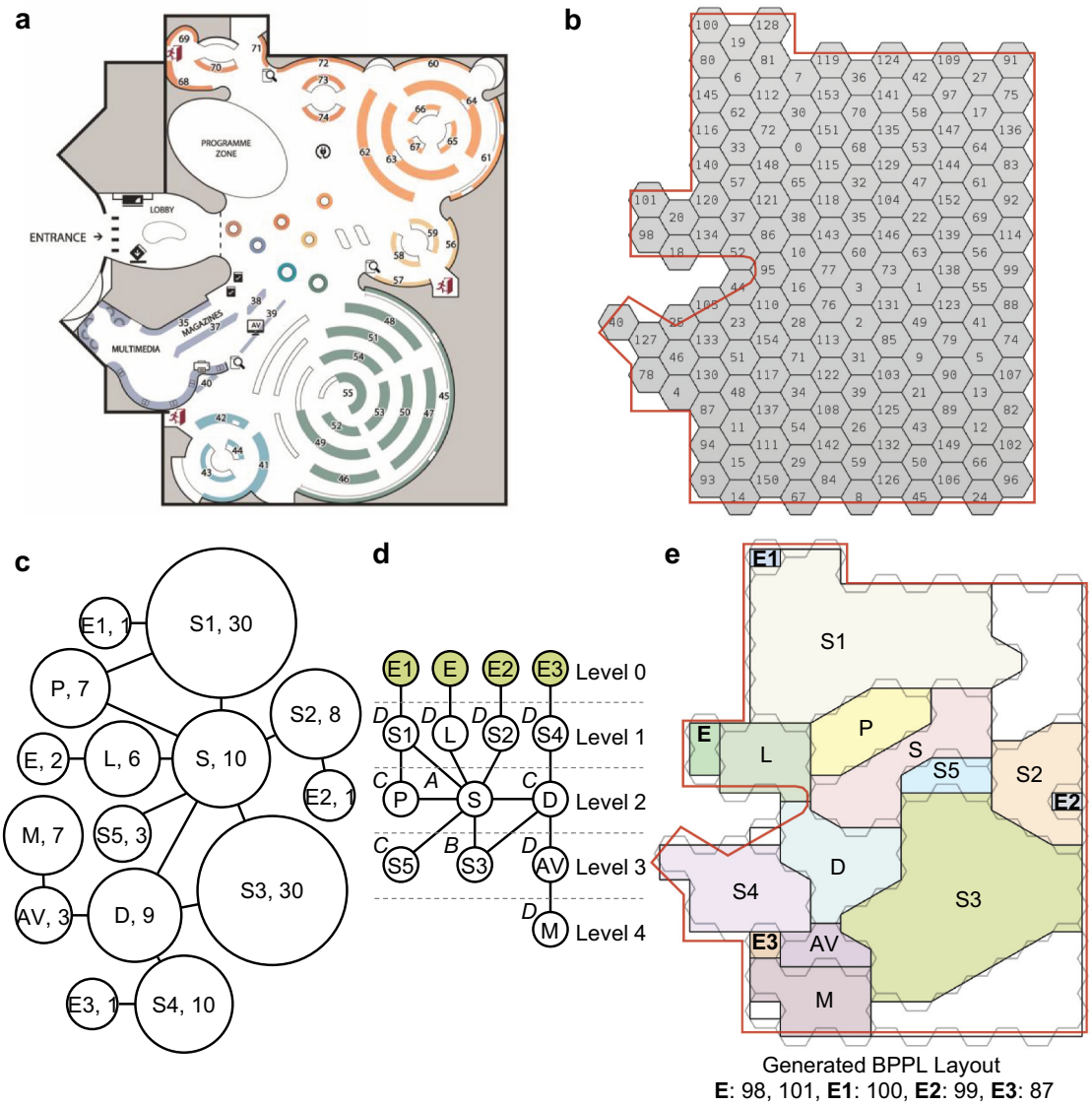

Figure 10. Adapted design parameters of Bukit Panjang Public library (BPPL) from which a layout is generated. a) the current real-world floorplan of BPPL. b) an approximation of the design boundary that is discretised into hexagonal cells. c) the adapted bubble diagram, d) organised classified into different levels and types after selecting the starting cells for the entrance and three exits. e) generated layout for BPPL with post-processing.

staff offices and storage. Here we show a layout that has been successfully generated together with post-processing (Figure 10e), adhering to all the adapted input parameters. This exploratory case study demonstrates how the proposed procedural model can accommodate user-defined inputs such as specifying the positions of key spaces within a fixed boundary. It illustrates the potential of assisting designers with real-world spatial planning challenges by bridging the gap between algorithmic generation and practical usability.

To further validate the feasibility of this method in actual projects, it will be beneficial to collaborate with architects and designers in future to conduct usability studies to explore aspects such as design intuitiveness and aesthetic preferences. Through such evaluations, we can examine how well the generated layouts align with real-world design expectations and how the design process can be further streamlined in terms of user control and interfacing. User feedback can help identify potential areas for improvement, such as refining the generation rules for greater flexibility and user control so as to ensure that the generated layouts do not only meet the user-defined spatial constraints but also qualitative or aesthetic design considerations.

While the current procedural model has limitations, it offers significant advantages over approaches based on machine learning (ML) for spatial layout planning. A key strength of procedural modelling lies in its ability to amplify data (Amato 2017; Cogo et al. 2023; Smelik et al. 2014). This is in contrast to ML models which require large, high-quality data sets for training and also often struggle to generate solutions that cater to scenarios outside the scope of their training data (Aalaei et al. 2023; Chaillou 2020; Ko et al. 2023; Nauata et al. 2020, 2021). Procedural modelling eliminates the dependency on such data sets, hence making it advantageous in contexts where design data is limited. Another critical advantage of the proposed method is its strict adherence to user-defined constraints, which are the adjacency, area, and design boundary in this work. The rule-based nature of the proposed procedural algorithms ensure that user inputs and requirements are reflected directly in the generated layouts. As the rules and algorithms driving the layout generation process are explicitly defined, a procedural approach is transparent and more interpretable compared to the "black box" nature of many ML models (Bayraktar Sari and Jabi 2024; Chen et al. 2022; Lipton 2018). This flexibility has the potential to be especially valuable in architectural design processes where iterative feedback and adjustments can guide the fulfilment of various functional and aesthetic goals.

The proposed procedural model introduces algorithms based on CA and graph theory, allowing it to address the basic objectives of spatial layout planning. By utilizing a hexagonal grid system and ordered generation of spaces, the model handles adjacencies well, ensuring that the layouts generated fit within given design boundaries while achieving their desired allocation of area. The incorporation of the post-processing step for transforming the hexagonal layouts into more rectangular forms also aligns better with conventional architectural designs while preserving the integrity of the given initial constraints. This combination of flexibility, precision, and adaptability therefore bridges automation and user agency, allowing designers to influence the generation process while benefitting from algorithmic efficiency. Therefore, we believe that the proposed approach is an ideal tool for exploring diverse layout possibilities for spatial layout planning.

A future direction will be to expand the model's capabilities to address more objectives of spatial layout planning. This could involve enhancing the CA algorithms with space syntax metrics such as visibility and circulation which have been recognised for optimising layouts for traffic flow and wayfinding (Keleş, Takva, and Çakıcı 2025; Natapov et al. 2015; Okhoya et al. 2022; Pilechiha et al. 2020; Pouyan, Ghanbaran, and Shakibamanesh 2021; Weber, Mueller, and Reinhart 2022). Proper circulation will ensure that spaces are accessible and movement within the space is logical and efficient to reduce unnecessary detours or congestion (Natapov et al. 2020; Penn and Turner 2002; Shekhawat et al. 2023). For instance, proper circulation in a museum would involve creating logical pathways between and within galleries for more intuitive flow. On the other hand, an analysis of the visibility of a space can be performed using isovists or what is defined as fields of vision from a specific point in space (Benedikt 1979). Visibility graph analyses make use of isovists to assess how much of a space is visually accessible from key points (R. Li and Klippel 2010; Turner et al. 2001). Optimised visibility helps people to orient themselves and locate important landmarks, improving wayfinding within the space would thereby also enhance the overall user experience as they navigate the space (Jamshidi and Pati 2021; Lynch 2008; Turner et al. 2001; van Nes and Yamu 2021). Currently, the algorithms have been developed to govern the layout formation based on adjacency constraints. Future modifications could integrate rules that account for pedestrian flow needs or visibility values to improve navigation and spatial perception within the generated layouts. Accounting for both circulation and visibility in the spatial layout planning process has the potential to significantly enhance user experience in public spaces, such as museums, transport hubs, or airports where smooth traffic flow and clear navigation are critical. For example, effective circulation in a museum can help to ensure seamless pathways between galleries, and optimised visibility can aid visitor navigation or highlight key exhibits.

The grid-based system of the proposed procedural model offers a natural platform for integrating these space syntax metrics. Visibility graphs can be constructed by counting the number of intervisible cells and calculating visibility values for each cell in the space. Using these visibility values, doorways and walls can then be strategically positioned to create or obstruct sightlines as specified by the designer. Hence, a potential improvement to the current algorithms would be to adapt the rule set to account for intervisible cells, allowing layouts to evolve based on sightline requirements and constraints. Circulation can potentially be optimised as another extension of the model through integration of agent-based modelling (ABM) and CA. ABM is a method that simulates the movement and behaviour of individual agents within a given space (Feng et al. 2016; Song and Whitehead 2019) and can provide insights into pedestrian flow, making it a valuable tool for improving circulation in spatial layout planning. Simulating movement patterns can aid in refining the arrangement of spaces within the spatial layout to alleviate any identified potential bottlenecks and improve traffic flow. In addition, CA, which is used as the foundation of the proposed spatial layout planning model, has been shown in the literature as floor field models to simulate pedestrian dynamics and evacuation (Khashoggi et al. 2018; Yang, Wang, and Qin 2015). These models study movement patterns, congestion, and evacuation efficiency (Khashoggi et al. 2018; Yang, Wang, and Qin 2015), demonstrating their potential to enhance circulation and wayfinding for spatial layouts. This therefore suggests that rules can be expanded to simulate and plan layouts with more regulated movement patterns in future. Perhaps a hybrid approach of ABM and CA, utilising simulated movement to adjust layout generation in real time, may be explored to tackle challenges in planning for good circulation. Such enhancements can further broaden the capabilities of the current procedural model to tackle more complex design challenges.

5. Conclusion

This study introduces a procedural generative design model as a robust alternative to machine learning (ML)-based methods for spatial layout planning. In contrast to ML approaches which often require large, high-quality datasets, the proposed model amplifies data and operates independently of extensive training data, while still generating layout variations that satisfy user-defined constraints. Through integrating cellular automata (CA) and graph theory, the model employs three algorithms developed here to ensure that layouts meet key requirements of area, adjacency, and design boundary. The use of a hexagonal grid system enhances adjacency fulfilment, and an additional post-processing step refines the layouts into more rectangular and orthogonal forms. The model's versatility is demonstrated through three mock layouts of public spaces and a real-world case study. Each case presents layouts generated with distinct user inputs. While the current approach effectively addresses primary spatial layout objectives, future enhancements to the CA algorithms could broaden the model's applicability by incorporating visibility and circulation analyses using space syntax theory and agent-based modelling.

In summary, this work advances computational design for spatial layout planning by offering a flexible, transparent, and data-independent alternative. We believe that its adaptability and flexibility will be especially valuable in context-sensitive design processes in engineering, architecture, and urban planning.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Aalaei, M., M. Saadi, M. Rahbar, and A. Ekhlassi. 2023. "Architectural Layout Generation Using a Graph-Constrained Conditional Generative Adversarial Network (GAN)." *Automation in Construction* 155:105053. https://doi.org/10.1016/j.autcon.2023.105053.

Adamatzky, A. I. 1996. "Computation of Shortest Path in Cellular Automata." *Mathematical and Computer Modelling* 23 (4): 105–113. https://doi.org/10.1016/0895-7177(96)00006-4.

Amato, A. 2017. "Procedural Content Generation in the Game Industry." In *Game Dynamics*, edited by O. Korn and N. Lee, 15–25. Cham: Springer.

Ayegba, P., J. Ayoola, E. Asani, and A. Okeyinka. 2020. "A Comparative Study of Minimal Spanning Tree Algorithms." In *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, edited by J. O. Fatokun, A. E. Okeyinka, O. O. Awe, J. B. Odili, V. U. Nwankwo, W. A. Akanni and J. O. Babayemi, 1–4. IEEE.

Bahrehmand, A., T. Batard, R. Marques, A. Evans, and J. Blat. 2017. "Optimizing Layout Using Spatial Quality Metrics and User Preferences." *Graphical Models* 93:25–38. https://doi.org/10.1016/j.gmod.2017.08.003.

Batty, M., H. Couclelis, and M. Eichen. 1997. "Urban Systems as Cellular Automata." *Environment & Planning: B, Planning & Design* 24 (2): 159–164. https://doi.org/10.1068/b240159.

Bayraktar Sari, A. O., and W. Jabi. 2024. "Architectural Spatial Layout Design for Hospitals: A Review." *Journal of Building Engineering* 97:110835. https://doi.org/10.1016/j.jobe.2024.110835.

Benedikt, M. L. 1979. "To Take Hold of Space: Isovists and Isovist Fields." *Environment & Planning: B, Planning & Design* 6 (1): 47–65. https://doi.org/10.1068/b060047.

Calixto, V., and G. Celani. 2015." A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014." *XIX Congress of the Iberoamerican Society of Digital Graphics*, edited by A. T. C. Pereira and R.

T. Pupo, 662–671. Vol. 2." Florianópolis/SC, Brazil: Blucher Design Proceedings.

Chaillou, S. 2020. "Archigan: Artificial Intelligence x Architecture." In *Architectural Intelligence*, edited by P. F. Yuan, M. Xie, N. Leach, J. Yao and X. Wang, 117–127. Singapore: Springer.

Chen, V., J. Li, J. S. Kim, G. Plumb, and A. Talwalkar. 2022. "Interpretable Machine Learning: Moving from Mythos to Diagnostics." *Communications of the ACM* 65 (8): 43–50. https://doi.org/10.1145/3546036.

Cogo, E., E. Krupalija, I. Prazina, Š. Bećirović, V. Okanović, S. Rizvić, and R. T. Mulahasanović. 2023. "A Survey of Procedural Modelling Methods for Layout Generation of Virtual Scenes." *Computer Graphics Forum* 43 (1): e14989. https://doi.org/10.1111/cgf.14989.

Dijkstra, E. W. 1959. "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* 1 (1): 269–271. https://doi.org/10.1007/BF01386390.

Feng, T.; L.-F. Yu; S.-K. Yeung; K. Yin; K. Zhou. 2016. "Crowd-Driven Mid-Scale Layout Design." *ACM Transactions on Graphics* 35 (4): 1–14. https://doi.org/10.1145/2897824.2925894.

Hernández Encinas, L., S. Hoya White, A. Martín Del Rey, and G. Rodríguez Sánchez. 2007. "Modelling Forest Fire Spread Using Hexagonal Cellular Automata." *Applied Mathematical Modelling* 31 (6): 1213–1227. https://doi.org/10.1016/j.apm.2006.04.001.

Hu, R., Z. Huang, Y. Tang, O. Van Kaick, H. Zhang, and H. Huang. 2020. "Graph2Plan: Learning Floorplan Generation from Layout Graphs." *ACM Transactions on Graphics* 39 (4): 118:1–118:14. https://doi.org/10.1145/3386569.3392391.

Jamshidi, S., and D. Pati. 2021. "A Narrative Review of Theories of Wayfinding Within the Interior Environment." *HERD: Health Environments Research & Design Journal* 14 (1): 290–303. https://doi.org/10.1177/1937586720932276.

Jiang, S., M. Wang, and L. Ma. 2023. "Gaps and Requirements for Applying Automatic Architectural Design to Building Renovation." *Automation in Construction* 147:104742. https://doi.org/10.1016/j.autcon.2023.104742.

Jowers, I., C. Earl, and G. S. Stiny. 2019. "Shapes, Structures and Shape Grammar Implementation." *Computer-Aided Design* 111:80–92. https://doi.org/10.1016/j.cad.2019.02.001.

Keleş, B. N., Ç. Takva, and F. Z. Çakıcı. 2025. "Accessibility Analysis of Public Buildings with Graph Theory and the Space Syntax Method: Government Houses." *Journal of Asian Architecture & Building Engineering* 24 (1): 199–213. https://doi.org/10.1080/13467581.2023.2292083.

Khashoggi, A., S. Alyas, L. Abduljawad, S. Halawani, and F. Haron. 2018. "Spatial Planning for Effective Evacuation Using Floor Field Cellular Automata Model." In *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, 1–6. IEEE.

Kim, K., and M. Cho. 2020. "Development of the Layout Method for a High-Rise Housing Complex Using Parametric Algorithm." *Journal of Asian Architecture & Building Engineering* 19 (1): 30–47. https://doi.org/10.1080/13467581.2019.1697273.

Ko, J., B. Ennemoser, W. Yoo, W. Yan, and M. J. Clayton. 2023. "Architectural Spatial Layout Planning Using Artificial Intelligence." *Automation in Construction* 154:105019. https://doi.org/10.1016/j.autcon.2023.105019.

Kruskal, J. B. 1956. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem." *Proceedings of the American Mathematical Society* 7 (1): 48–50. https://doi.org/10.1090/S0002-9939-1956-0078686-7.

Li, J., J. Yang, A. Hertzmann, J. Zhang, and T. Xu. 2019. "LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators." arXiv:1901.06767. https://doi.org/10.48550/arXiv.1901.06767.

Li, R., and A. Klippel. 2010. "Using Space Syntax to Understand Knowledge Acquisition and Wayfinding in Indoor Environments." In *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, edited by F. Sun, Y. Wang, J. Lu, B. Zhang, W. Kinsner and L. A. Zadeh, 302–307. Beijing, China: IEEE.

Li, Y., M. Chen, Z. Dou, X. Zheng, Y. Cheng, and A. Mebarki. 2019. "A Review of Cellular Automata Models for Crowd Evacuation." *Physica A: Statistical Mechanics and Its Applications* 526:120752. https://doi.org/10.1016/j.physa.2019.03.117.

Liao, W., X. Lu, Y. Huang, Z. Zheng, and Y. Lin. 2021. "Automated Structural Design of Shear Wall Residential Buildings Using Generative Adversarial Networks." *Automation in Construction* 132:103931. https://doi.org/10.1016/j.autcon.2021.103931.

Lipton, Z. C. 2018. "The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery." *Queue* 16 (3): 31–57. https://doi.org/10.1145/3236386.3241340.

Liu, J., Z. Qiu, L. Wang, P. Liu, G. Cheng, and Y. Chen. 2024. "Intelligent Floor Plan Design of Modular High-Rise Residential Building Based on Graph-Constrained Generative Adversarial Networks." *Automation in Construction* 159:105264. https://doi.org/10.1016/j.autcon.2023.105264.

Lobos, D., and D. Donath. 2010. "The Problem of Space Layout in Architecture: A Survey and Reflections." *Arquitetura Revista* 6 (2): 136–161. https://doi.org/10.4013/arq.2010.62.05.

Lynch, K. 2008. *The Image of the City*. Vol. 33. Cambridge, Mass: Publication of the Joint Center for Urban studies; M.I. T. Press.

Natapov, A., S. Kuliga, R. C. Dalton, and C. Hölscher. 2015. "Building Circulation Typology and Space Syntax Predictive Measures." In *Proceedings of the 10th International Space Syntax Symposium*, edited by K. Kayvan, 30. London, UK: Space Syntax Laboratory, The Bartlett School of Architecture, University College London.

Natapov, A., S. Kuliga, R. C. Dalton, and C. Hölscher. 2020. "Linking Building-Circulation Typology and Wayfinding: Design, Spatial Analysis, and Anticipated Wayfinding Difficulty of Circulation Types." *Architectural Science Review* 63 (1): 34–46. https://doi.org/10.1080/00038628.2019.1675041.

National Library Board, Singapore. 2025. Bukit Panjang Library Floor Plan Adults' & Teens' Zones. Accessed November 12, 2024. https://www.nlb.gov.sg/main/visit-us/our-libraries-and-locations/libraries/bukit-panjang-public-library.

Nauata, N., K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa. 2020. "House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation." In *Computer Vision–ECCV 2020: 16th European Conference*, edited by A. Vedaldi, H. Bischof, T. Brox and J. M. Frahm, 162–177. Glasgow, UK: Springer International Publishing.

Nauata, N., S. Hosseini, K.-H. Chang, H. Chu, C.-Y. Cheng, and Y. Furukawa. 2021. "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*, edited by M. S. Brown, R.

Sukthankar, T. Tan and L. Zelnik, 13632–13641. Nashville, TN: IEEE.

Ng, W. K. 2017. Revamped Bukit Panjang Library to Reopen on July 1, Singapore. *The Straits Times*.

Nugraha, A. T., B. J. Waterson, S. P. Blainey, and F. J. Nash. 2021. "On the Consistency of Urban Cellular Automata Models Based on Hexagonal and Square Cells." *Environment & Planning B: Urban Analytics & City Science* 48 (4): 845–860. https://doi.org/10.1177/2399808319898501.

Okhoya, V. W., M. Bernal, A. Economou, N. Saha, R. Vaivodiss, T.-C. K. Hong, and J. Haymaker. 2022. "Generative Workplace and Space Planning in Architectural Practice." *International Journal of Architectural Computing* 20 (3): 645–672. https://doi.org/10.1177/14780771221120580.

Omer, Y., A. Spinks, and C. Bednarz. Re-Logic. 2011. Terraria505 Games.

Paulino, D. M. S., H. Ligler, and R. Napolitano. 2023. "A Grammar-Based Approach for Generating Spatial Layout Solutions for the Adaptive Reuse of Sobrado Buildings." *Buildings* 13 (3): 722. https://doi.org/10.3390/buildings13030722.

Penn, A., and A. Turner. 2002. "Space Syntax Based Agent Simulation." In *Pedestrian and Evacuation Dynamics*, edited by M. Schreckenberg and S. D. Sharma, 99–114. Spinger-Verlag.

Persson, M., and J. Bergensten. Mojang Studios. 2011. Minecraft. Mojang Studios.

Pilechiha, P., M. Mahdavinejad, F. Pour Rahimian, P. Carnemolla, and S. Seyedzadeh. 2020. "Multi-Objective Optimisation Framework for Designing Office Windows: Quality of View, Daylight and Energy Efficiency." *Applied Energy* 261:114356. https://doi.org/10.1016/j.apenergy.2019.114356.

Pouyan, A. E., A. Ghanbaran, and A. Shakibamanesh. 2021. "Impact of Circulation Complexity on Hospital Wayfinding Behavior (Case Study: Milad 1000-Bed Hospital, Tehran, Iran)." *Journal of Building Engineering* 44:102931. https://doi.org/10.1016/j.jobe.2021.102931.

Prim, R. C. 1957. "Shortest Connection Networks and Some Generalizations." *Bell System Technical Journal* 36 (6): 1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x.

Purho, P., O. Harjola, A. Teikari, and A. Tiihonen. Nolla Games. 2020. Noita. Nolla Games.

Shekhawat, K., R. Lohani, C. Dasannacharya, S. Bisht, and S. Rastogi. 2023. "Automated Generation of Floorplans with Non-Rectangular Rooms." *Graphical Models* 127:101175. https://doi.org/10.1016/j.gmod.2023.101175.

Sirakoulis, G. C., I. Karafyllidis, and A. Thanailakis. 2000. "A Cellular Automaton Model for the Effects of Population Movement and Vaccination on Epidemic Propagation." *Ecological Modelling* 133 (3): 209–223. https://doi.org/10.1016/S0304-3800(00)00294-5.

Smelik, R. M., T. Tutenel, R. Bidarra, and B. Benes. 2014. "A Survey on Procedural Modelling for Virtual Worlds." *Computer Graphics Forum* 33 (6): 31–50. https://doi.org/10.1111/cgf.12276.

Song, A., and J. Whitehead. 2019. "TOWNSIM: Agent-Based City Evolution for Naturalistic Road Network Generation." In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, edited by S. Deterding, F. Khosmood, J. Pirker and T. Apperley, 1–9. San Luis Obispo, CA: ACM.

Stiny, G., and J. Gips. 1972. "Shape Grammars and the Generative Specification of Painting and Sculpture." In *Proceedings: C*, edited by V. Freiman, 1460–1465. Vol. 71. Ljubljana, Yugoslavia: *Information Processing*.

Turner, A., M. Doxa, D. O'Sullivan, and A. Penn. 2001. "From Isovists to Visibility Graphs: A Methodology for the Analysis of Architectural Space." *Environment & Planning: B, Planning & Design* 28 (1): 103–121. https://doi.org/10.1068/b2684.

Ulam, S. 1952. "Random Processes and Transformations." *Proceedings of the International Congress of Mathematicians* 2:264–275.

Valiente, G. 2021. *Algorithms on Trees and Graphs: With Python Code; Texts in Computer Science*. Cham: Springer International Publishing.

van der Linden, R., R. Lopes, and R. Bidarra. 2014. "Procedural Generation of Dungeons." *IEEE Transactions on Computational Intelligence and AI in Games* 6 (1): 78–89. https://doi.org/10.1109/TCIAIG.2013.2290371.

van Nes, A., and C. Yamu. 2021. *Introduction to Space Syntax in Urban Studies*. Cham: Springer International Publishing.

von Neumann, J. 1966. *Theory of Self-Reproducing Automata*. Edited by A. W. Burks. Urbana and London: University of Illinois Press.

Wang, J., J. Zhou, and S. Zhang. 2025. "Applying Shape Grammar and BIM for Generating the Mass Modular Housing Design in Shanlichenjia Village, Zhaoyuan, China." *Journal of Asian Architecture & Building Engineering* 24 (1): 178–198. https://doi.org/10.1080/13467581.2023.2292081.

Wang, X.-Y., and K. Zhang. 2020. "Generating Layout Designs from High-Level Specifications." *Automation in Construction* 119:103288. https://doi.org/10.1016/j.autcon.2020.103288.

Weber, R. E., C. Mueller, and C. Reinhart. 2022. "Automated Floorplan Generation in Architectural Design: A Review of Methods and Applications." *Automation in Construction* 140:104385. https://doi.org/10.1016/j.autcon.2022.104385.

Yang, X., B. Wang, and Z. Qin. 2015. "Floor Field Model Based on Cellular Automata for Simulating Indoor Pedestrian Evacuation." *Mathematical Problems in Engineering* 2015 (1): 1–10. https://doi.org/10.1155/2015/820306.

Zhao, C.-W., J. Yang, and J. Li. 2021. "Generation of Hospital Emergency Department Layouts Based on Generative Adversarial Networks." *Journal of Building Engineering* 43:102539. https://doi.org/10.1016/j.jobe.2021.102539.

Zhao, R., Y. Zhai, L. Qu, R. Wang, Y. Huang, and Q. Dong. 2021. "A Continuous Floor Field Cellular Automata Model with Interaction Area for Crowd Evacuation." *Physica A: Statistical Mechanics and Its Applications* 575:126049. https://doi.org/10.1016/j.physa.2021.126049.

Zhong, G., G. Zhai, and W. Chen. 2023. "Evacuation Simulation of Multi-Story Buildings During Earthquakes Based on Improved Cellular Automata Model." *Journal of Asian Architecture & Building Engineering* 22 (2): 1007–1027. https://doi.org/10.1080/13467581.2022.2070491.