Journal Pre-proof

Automated Building Layout Generation: Implementation and Comparison of Streamer Early Design Configurator and SDaC Layout Designer

Yingcong Zhong, Steffen Hempel, Andreas Geiger, Karl-Heinz Haefele, Veit Hagenmeyer

PII: S2352-7102(24)01731-5

DOI: https://doi.org/10.1016/j.jobe.2024.110163

Reference: JOBE 110163

To appear in: *Journal of Building Engineering*

Received Date: 22 November 2023

Revised Date: 2 July 2024

Accepted Date: 6 July 2024

Please cite this article as: Y. Zhong, S. Hempel, A. Geiger, K.-H. Haefele, V. Hagenmeyer, Automated Building Layout Generation: Implementation and Comparison of Streamer Early Design Configurator and SDaC Layout Designer, *Journal of Building Engineering*, https://doi.org/10.1016/j.jobe.2024.110163.

Automated Building Layout Generation: Implementation and Comparison of Streamer Early Design Configurator and SDaC Layout Designer

Yingcong Zhong[a], Steffen Hempel[a], Andreas Geiger[a], Karl-Heinz Haefele[a] and Veit Hagenmeyer[a]

[a] **Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, Eggenstein-Leopoldshafen, 76344**

Abstract

When designing new buildings, various constraints must be taken into account. These constraints encompass factors such as the overall building size, the number, function, and size of required spaces, and the surrounding neighborhood's existing buildings, infrastructure, and natural environment. Presently, these constraints are predominantly considered in a formal manner during state-of-the-art design processes. This paper proposes a novel approach that aims to formalize space requirements and utilizes geospatial information surrounding the building site. This approach facilitates the generation of different design alternatives during the early design phase. The advantages of this approach are numerous. By representing space requirements in a machine-readable format, it becomes possible to semi-automatically generate various design alternatives for a simplified building envelope and a basic space layout.

Particularly in large areas such as hospital districts or industrial sites, multiple options for the location of a new building can be explored. Leveraging geospatial information allows designers to assess whether a specific design variant can be practically implemented at a given location. This supports the decision-making process in selecting the most promising early design model, which in turn informs subsequent design steps. This paper introduces two methods, namely Streamer Early Design Configurator and SDaC Layout Designer, to generate multiple initial building floor plans. These drafts are enriched with additional data, including placement information, volume details, and space utilization. Ultimately, the floor plans are exported into three-dimensional objects and exported in the IFC format. This comprehensive step paves the way for more efficient and informed building design, promoting greater flexibility and adaptability in the early stages of the architectural process.

Keywords: Automated layout generation, BIM generation, Evolutionary strategies, Mathematical programming

1. Introduction

The traditional layout design in the architecture and construction industry is usually accomplished manually by architects, while computers mostly play the role as modeling, rendering and printing tool. However, the research of computer-aided layouts design has started since the 1960s [31], and many approaches and applications have been developed based on various design mechanisms. In the field of building layout design, there are multiple factors that need to be considered at the same time, resulting in increasing complexity of calculation. These factors include the geometric and topological information of spaces inside the building, as well as the environment and infrastructures around the building. Generative design frees the designers from the "trial and error" process, and with the application of artificial intelligence, it would also improve productivity, safety and quality in not only for the layout designing, but also in other construction processes [1].

Building information modeling (BIM) provides a common data environment for visualization, collaboration, cost estimation, energy performance optimization and facility management. The early concept of BIM has been proposed since the 1970s, aiming to comprehensively describe the construction processes in a digital way [8, 9]. The sooner BIM is applied in the life cycle of a building project, the more it helps to improve the efficiency and accuracy of the design process while reducing costs and errors. It also provides valuable information that can be used throughout the entire life cycle of a building, from design to construction to operation and maintenance. Therefore, the automatic generation of building layouts and export of BIM models become significantly important in the architect and construction industry.

In this article two methods are presented for the automatic generation of building layouts, the Streamer Early Design Configurator (EDC) and the SDaC Layout Designer. This paper starts with a brief review of the state of the art, introducing different methods used for layout generation. Second, the Streamer EDC is presented with its workflow and working principle, the evolutionary algorithms. Then the SDaC Layout Designer is introduced as well as how its mathematical model is built based on the program of requirements (PoR). To test and evaluate both methods, a list of PoR is used in both developed tools and layouts are generated, which are also exported as openBIM data (IFC). At the end both methods are compared in the summary and the future research possibilities are discussed.

2. State-of-the-art automated layout generation

Based on the procedure, the problem of layout generation can be classified into two types: Outside-In and Inside-Out [5]. As is shown in figure 1, Outside-In means designing the interior layouts within a given space domain, which is limited in size and geometry. Inside-Out means generating the layouts with spaces that fulfill their requirements, and the building outline is then determined by the combination of the generated spaces. Most research focus on the Outside-In design procedure, and so do the approaches that are discussed in this paper. In order to allocate space to specific domains, various algorithms such as dense packing or subdivision are proposed [16], combined with different algorithms such as evolutionary strategies.

The research of layout generations has covered a variety of approaches based on different generating principles, including physically based methods, mathematical optimization, graph-theory aided, cell assignment, space splitting, occupant-trace and machine learning [7], etc. Despite different principles among these methods, they are commonly combined in the research for layout generation. The research works have different focus on the generating process, such as energy performance, travelling distances between spaces, view impedance or space utilization.

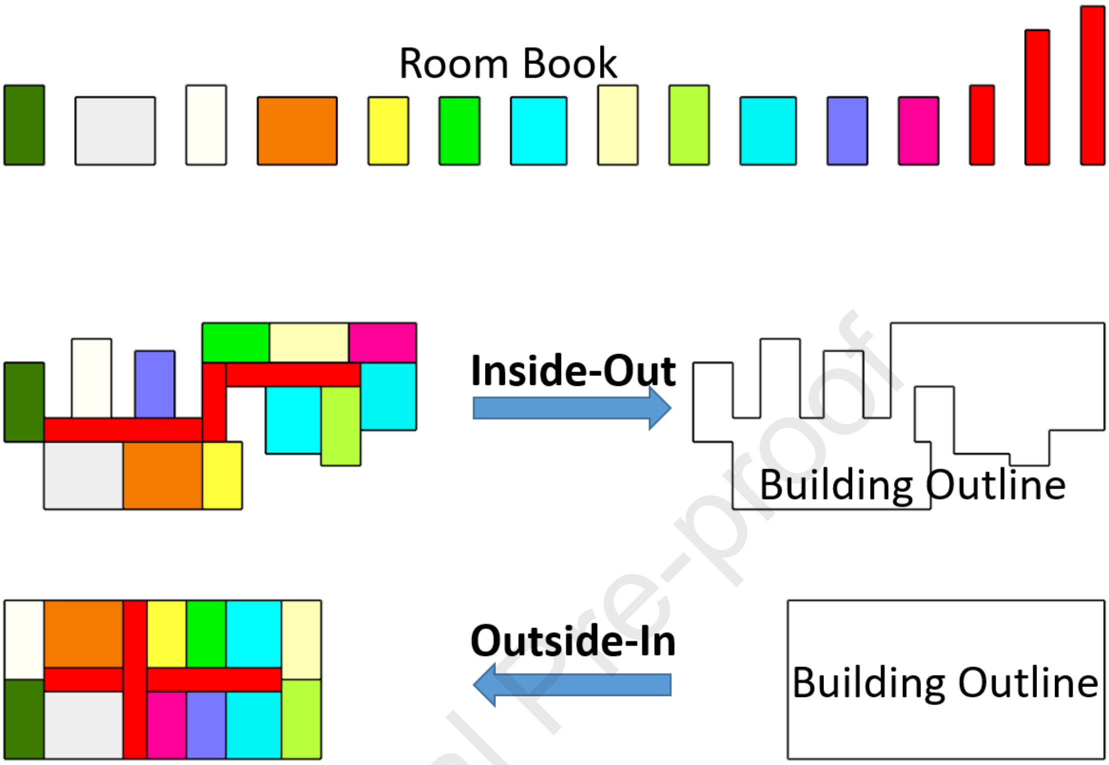

Figure 1: Example of outside-in and inside-out design mode

2.1. Mathematical programming

For the mathematical programming method, the layout generation problems are firstly formulated with mathematical language. Space layouts are represented with geometric parameters, while their adjacency and functionalities are transformed into constraints. The mathematical optimization problem can be solved with the help of different solvers or tool sets, and the results are turned into layouts for further usage. In [20] an optimization model of the quantifiable aspects of architectural floorplan layout design is presented, which is solved by using evolutionary algorithms to make discrete decisions and do global search. Wu [32] introduces a mix-integer quadratic programming formulation to describe the layout generation problem, in which the requirements such as size, position and adjacency are transformed to constraints of the optimization problem. Egor [10] presents a quasi-evolutionary strategy to generate floor plans in an iterative manner based on the connectivity and adjacency between specific rooms.

2.2. Graph-theory aided

There is research focusing on the adjacency between spaces in the layout design. Such relationship is transformed into a topological graph, in which nodes represent spaces and edges represent connection. Such graph can be turned into 2D matrix for feasibility verification and arithmetic solution. With the solution of space adjacency, other methods would also be applied to complete the geometric design for layout generation. Shekhawat presents a tool to construct dimensioned layouts using graph-theoretical and optimization techniques in [26], in which the adjacency requirements are given either in an adjacency graph or in a dimensionless layout. Slusarczyk presents hierarchical graph-based data structures for representing design solutions together with graph grammars [27], in which the local formulas expressing design properties are transformed into equivalent graph requirements.

2.3. Cell assignment

The building geometry can be predefined and divided into cells, which are later assigned to specific spaces. The assignment is based on different rules or requirements such as space's dimension, position or its functionality, as well as the space's non-overlapping and shape-continuity. By rastering the building footprint polygon into 2D grids, Lopes proposes a room expansion algorithm in [17] to generate building layout that fulfills the adjacency and connectivity constraints. Herr [14] examines the adaptations that cellular automata are typically subjected to, when they are applied to architectural designing, and discusses the challenges and opportunities met by designers when employing and developing cellular automata as design tools.

2.4. Machine learning

Based on data sets of space layouts from real cases, machine learning models become more and more applicable for automated layout generation. Such model can learn the characteristics of real space layouts and the experience would be implicitly used for the generation process. Merrell [19] presents a Bayesian Network to structurally learn the adjacency of rooms, from which the results are further optimized for detailed design. Chaillou [4] trains generative adversarial networks (GAN) [12] with real cases floorplan pictures to generate building footprints, split rooms and place furniture step by step. Nauata [21] also trains GANs and uses topological graph diagram as input for geometric design of floorplans.

2.5. Overview

One of the challenges of layout generation is, as a user-driven task, the input from the user is inevitably required, otherwise it will be time consuming to pick up preferred results from thousands of computer-generated solutions. In addition, the results are expected to be vectorized data, so that they can be reused in other applications for further validation and simulation.

3. Streamer Early Design Configurator

The Streamer Early Design Configurator (EDC) was developed as part of the EU funded industry-driven collaborative research project STREAMER [13, 25, 28]. Goal of the STREAMER project was to increase energy efficiency of health-care buildings [2]. The EDC creates early design proposals for large buildings with many rooms with functional dependencies.

Functional relations between rooms are e.g., that rooms share resources and must be in close proximity. Therefore, the position of a room is strongly dependent on multiple parameters: National and international norms and standards, rules based on tacit knowledge and project specific requirements. These parameters are defined in a common rule set language.

The rooms used in the project are predefined in a program of requirements (PoR) which contains information such as amount, minimum area, and textual parameters specific to the STREAMER project (e.g., Room type, accessibility class, hygiene class ...) [29, 30].

The building where the rooms are placed is a predefined 2D representation. Two modes are available: A corridor placement mode, where empty rectangular building segments are filled with corridor layouts (see figure 2) which in turn are filled with the rooms from the PoR and a free space mode, where no restrictions exist for placing walls, but rooms are still arranged consecutively in a linear fashion within rectangular free spaces.

By utilizing an evolutionary algorithm, rooms are placed to generate a floor plan. The floor plan generates a fitness value that quantifies how effectively it meets the specified requirements. Possible floor plans are generated by randomly changing the position of rooms.

Only rectangular free space is supported, and only rectangular rooms are placed.

The floor plan with the best fitness can be exported as a BIM model.

3.1. Process description

The building hull is generated or imported. Generated building hulls can be used in corridor mode, where a corridor layout is placed and then filled with rooms, imported building hulls provide free spaces that are filled with rooms. Imported building hulls are often used in refurbishment scenarios.

After loading the PoR and the rule set [22], the algorithm is started and begins with an empty floor plan as shown in Figure 3.

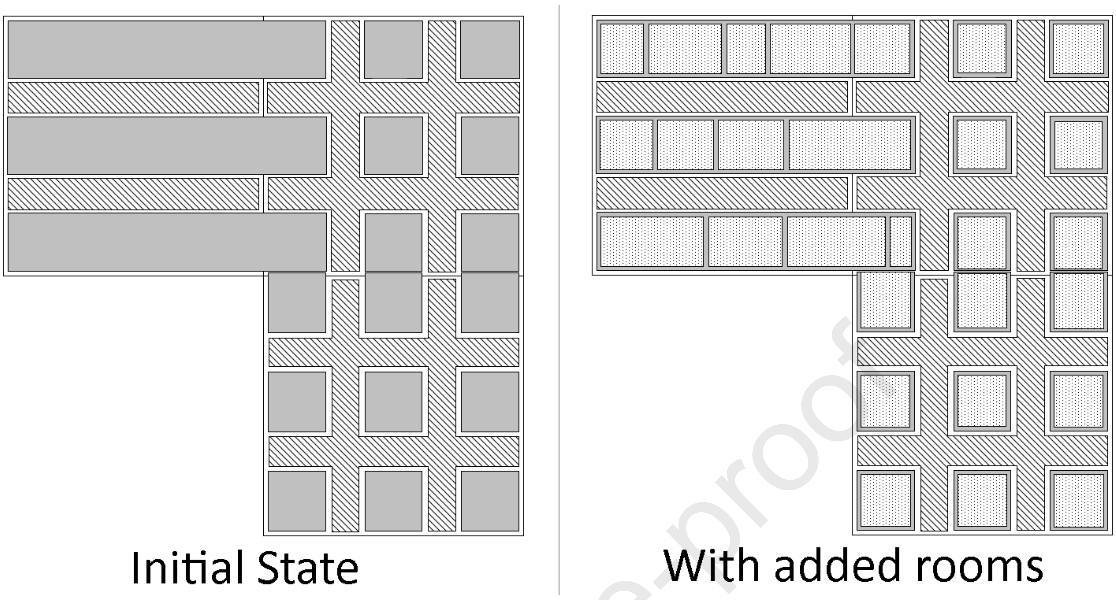

Figure 2: Example of corridor mode in the EDC, corridor layout hatched, resulting free space in gray, initial state without rooms left, added rooms right.

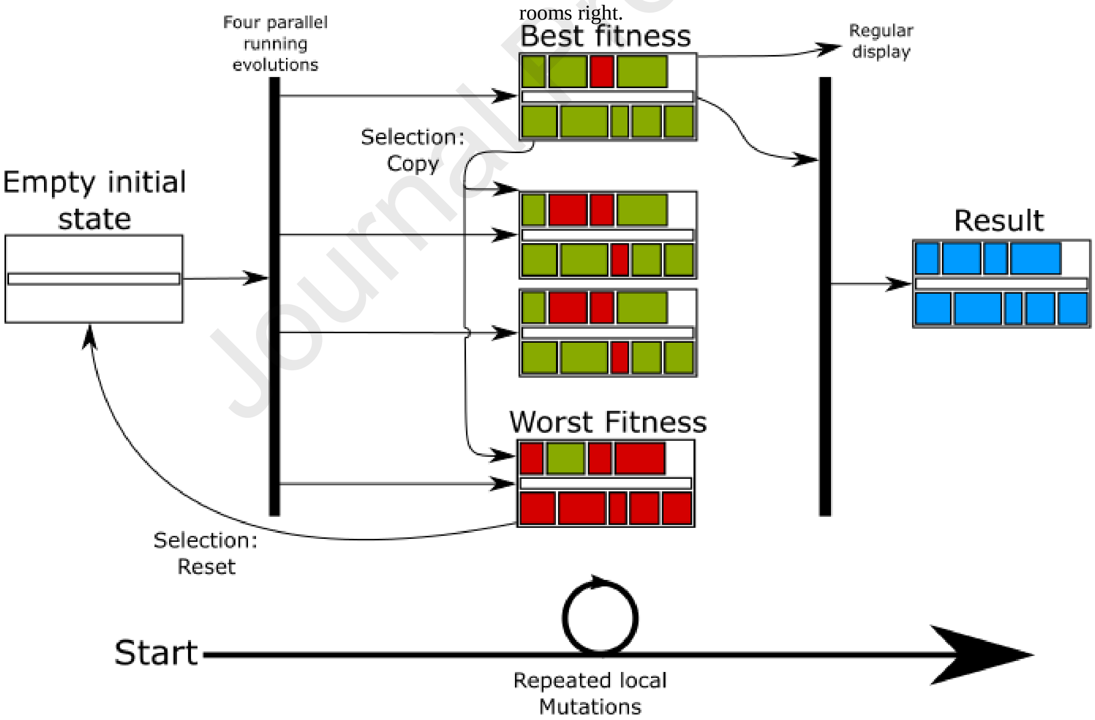

Figure 3: Layout generation process

156 The initial state is further processed in parallel. A certain number of mutations is applied
157 locally to each parallel process. After those mutations the selection takes place. All floor plans

with a below average fitness are cleared and restarted. In corridor mode these floor plans may receive a new corridor layout. The floor plans with above average fitness are replaced with the floor plan with the best fitness. The floor plan with the best fitness is also displayed graphically to the end user. The end user also decides if the displayed floor plan is sufficient and stops the processing. The processing can be resumed if the result is not sufficient. The current result can also be cloned, in order to develop multiple alternatives.

Mutation occurs by selecting a room either from a list of rooms that have not yet been inserted or, by removing a room from the floor plan. The selected room is then inserted in a random free space, either before, after or in between already inserted rooms in the selected free space. If the free space is empty and the minimum area of the room is larger than the area of the free space, the mutation fails. If the free space is already full, a random room is removed from the free space. If the room is still too large the mutation fails. If a room has been removed in favor of the first selected room, it is inserted with the aforementioned method. This results in a recursion that is done to a predefined depth before the mutation fails. If the mutation fails, the floor plan is reset to the valid state before starting the mutation.

3.2. Rule definition and implementation

The fitness of a floor plan is calculated from the sum of all fitness values generated from their corresponding rules. Rules also have a priority value assigned which is multiplied with the fitness value to give increase or decrease fitness according to priority.

Rules are defined in a domain specific language and loaded into the EDC.

Additionally, to user defined rules hard coded rules exist which generate bad fitness values if rooms are unused (not in the floor plan) or if rooms are not sized to a predefined length and width ratio.

User defined rules query a number of rooms into a set and apply a relation to them with predefined parameters. Unary rules apply the relation between all rooms in the same set. Binary rules apply their relation between rooms in two room sets.

Unary relations are:

- Cluster: All rooms must be connected.
- Cluster (same floor): All rooms must be on the same floor and connected.
- Distance to outer wall: All rooms must be at a minimum or maximum distance to the outer wall.
- Predefined floor: All rooms must be at the same floor.

Binary relations are:

- Minimum/maximum distance: All rooms from a set must be at a minimum/maximum distance from another set.
- Same/different floor: All rooms from a set must be on the same/a different floor than rooms from another set.
- (Partially) Overlapping below/above: All rooms from a set must be (partially) overlapping below/above another set.

3.2.1. Calculating fitness from rules

Calculating the fitness of a floor plan is executed by calculating the sum of all fitness values from the current layout states *V* multiplied with a priority value for each rule as seen in equation 1:

$$\mathrm{fitness_{global}}(\mathrm{V}) = \sum_{\mathrm{i}=0}^{\mathrm{number\ of\ rules}} \mathrm{fitness}(\mathrm{V_i}) \cdot \mathrm{priority_i} \quad (1)$$

There are three cases. Depending on the rule a different method is used to calculate the fitness value (see equation 2).

$$\mathrm{fitness}(v) = \begin{cases} \mathrm{fitness_{soft}}(v) \\ \mathrm{fitness_{hard}}(v) \\ \mathrm{fitness_{combined}}(v) \end{cases} \quad (2)$$

The better the rule is fulfilled, the better fitness it gets with the soft calculation method.

$$\mathrm{fitness_{soft}}(v) = \text{how good the rule is fulfilled} \quad (3)$$

The hard fitness (see equation 4) calculation uses a range defined in the rule. If the rule is evaluated to be in a certain range of target values the fitness is set to a constant best fitness value, or to a constant bad fitness value else.

$$\mathrm{fitness_{hard}}(v) = \begin{cases} \text{best fitness} & \text{if } v \text{ in range} \\ \text{bad fitness} & \text{else} \end{cases} \quad (4)$$

The combined fitness function (see equation 5) uses a combination of the soft and hard fitness calculation method. If the rule is evaluated to be in a certain range of target values, the soft calculation value is used. This means the fitness of this rule is at least good enough but a better result is possible. If the rule is evaluated to be outside of a certain range of target values the fitness is a constant bad fitness value.

$$\text{fitness}_{\text{combined}}(v) = \begin{cases} \text{fitness}_{\text{soft}}(v) & \text{if } v \text{ in range} \\ \text{bad fitness} & \text{else} \end{cases} \quad (5)$$

3.2.2. Example: Distance

Distance between Room A and Room B must be minimal and must be 50 m maximum.

The relation is *Distance between ... must be minimal and must be ... maximum*, queries are *Room A* and *Room B* and parameter is *50 m*.

A good fitness has been reached if distance between *Room A* and *Room B* is smaller than 50 m, and will become better the smaller the distance is. Bad fitness would be, if the distance between *Room A* and *Room B* were greater than 50 m.

Distance calculation between rooms is simply calculated between the edges of the rooms.

3.2.3. Example: Cluster

All rooms in Hygiene Class 1 should be clustered.

The query is *All rooms in Hygiene Class 1* and the relation is *should be clustered*. This example has no parameters.

The largest cluster group defines the fitness. The best fitness is reached if all rooms are in one cluster.

DBSCAN [11] is used to create clusters from the rooms. Rooms are connected if they are directly next to each other or on the other side of a corridor. If a room is connected to more than

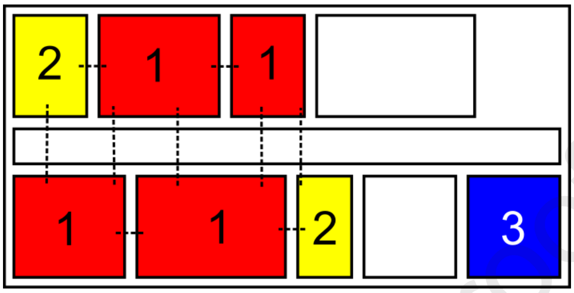

Figure 4: DBSCAN applied to rooms, red: core points. yellow: reachable points. blue: noise.

three other rooms, it is defined as a core room. If a room has less than three connections but is reachable through a core point it is part of their cluster. If a room has no neighbors, it is not part of a cluster. All rooms reachable from a core point without going over non-core points are part of the same cluster (see figure 4).

4. SDaC Layout Designer

The research project Smart Design and Construction (SDaC) aims to link the heterogeneous data in the construction industry, make different organizations work together on a platform, and discover the possibilities of artificial intelligence in architecture engineering and construction [23]. In this project a platform is built and maintained, on which various applications are provided to help on different processes through a construction's life cycle. SDaC Layout Designer aims to provide various building interior layouts that can be used by other applications [33]. The spaces of the interior layout are described in a tabular form and are transformed into a mathematical optimization problem. With the help of nonlinear programming solver, multiple solutions could be found, and each represents a possible layout. These solutions are then exported as IFC data, which can be read and edited in other CAD software.

4.1. Workflow of the program

To generate the interior layouts, the contour of the building is first drafted. Each floor of the building can be set with different parameters such as length, width and height. In the SDaC Layout Designer, the contour of each floor can be an axis-aligned polygon. The core part of the workflow is to create a room book or import one from other source. This room book contains the requirements for the layout design, can be edited in the program for different proposal generations, and will be saved as CSV format for later use. Once the building and the room book is defined, a mathematical programming problem is created for the layout generation. Every feasible solution of the optimization problem is acquired once it is found by the solver, until the optimization is paused or reaches the global optimum. Every solution that is found will be transformed to a 2D layout and visualized. After the optimization, the result could be exported to IFC file with 3D modelling data for further usage.

4.1.1. Drafting the building contour

The outline of each floor of the building can differ, but is always limited as axis-aligned polygon. To simplify the draft in this process, the user can define the length and width of a rectangle as the bounding box, and optionally select and remove one of the corners to make the outline into "L" shape. Figure 5 shows the user interface of how to draft the building contour.

4.1.2. Definition of requirements

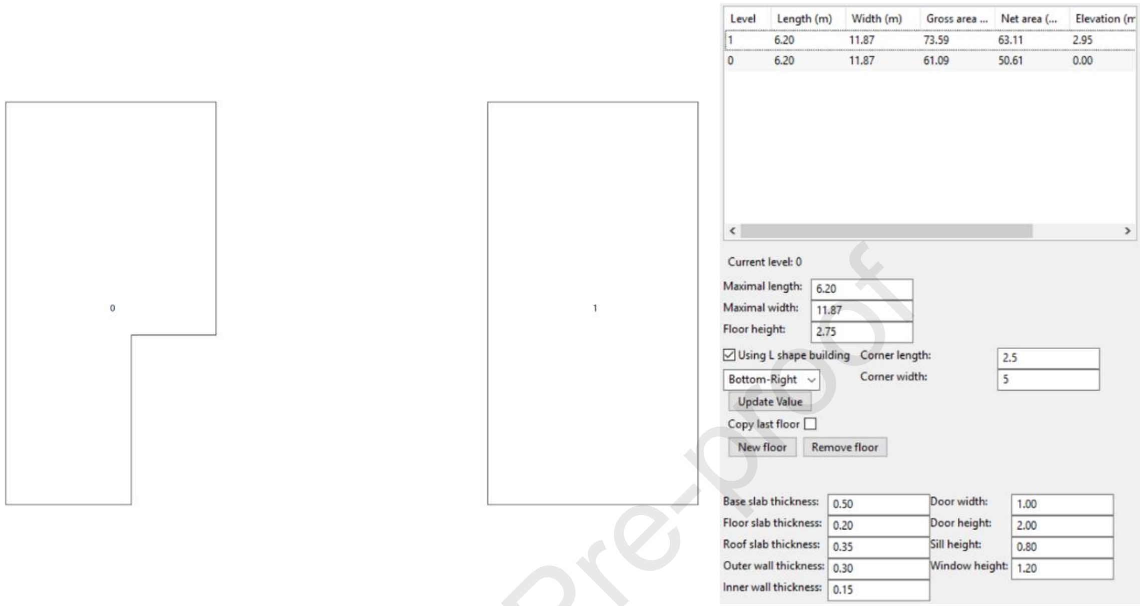

Figure 5: User interface for editing building contour

The program of requirements can either be imported from CSV data or directly created in the program. As is shown in figure 6, the user can define the dimension, position of a room, as well as its neighboring relationship with the other rooms. Figure 7 shows a topological graph, representing all the adjacency of the requirements.

4.2. Mathematical modeling of the layout design problem

A rectangle is used to represent the room in this early-design stage. The parametric description of each room with index i includes the continuous variables (x_i, y_i) denoting the bottom-left corner of the rectangle, (l_i, w_i) denoting the length and the width of the rectangle, and the binary variable f_{ik} denotes whether this room is on the k-th floor or not. These binary variables follow a basic constraint in equation 6:

$$\sum_{k=1}^{K} f_{ik} = 0 \quad (6)$$

k denotes the number of floors of the building.

In layout generation, one of the main objectives is to cover the buildable area of each floor as much as possible. Besides, to reach the user's requirement, the dimension of each room should be as close as possible to the user's input. It is possible to formulate multiple objectives in the optimization model, or to formulate them in one equation with user-preferred weight for each objective as described in equation 7:

$$\min \sum_{1}^{K} A(k) - \sum_{1}^{N} l_i w_i + \sum_{1}^{N} (l_i - l_{ti})^2 + (w_i - w_{ti})^2 \quad (7)$$

A(k) denotes the buildable area of each floor k and (l_{ti}, w_{ti}) represent the target dimension from the requirements.

The constraints of the optimization model are derived from the requirements of each room. These constraints are classified as general constraints and specific constraints. General constraints are considered independently of the room types or building types. These include the position, dimension and neighboring relationship of the rooms:

Maximal/Minimal area, length, width and aspect ratio

(l_{ti}, w_{ti}) are the target dimension which is used in the objective function. In practice the values of (l_i, w_i) could deviate, but should still be limited within an appropriate interval. In addition, in order to prevent a room becomes too narrow or too long, its aspect ratio can also be constrained.

Non-overlap

No rooms should overlap with each other. The constraint is described in equation 8.

$$\begin{cases} x_i \geq x_j + l_j - M(1-\theta_{ij}^R f_{ij}) \\ x_j \geq x_i + l_i - M(1-\theta_{ij}^L f_{ij}) \\ y_i \geq y_j + w_j - M(1-\theta_{ij}^T f_{ij}) \\ y_j \geq y_i + w_i - M(1-\theta_{ij}^B f_{ij}) \\ \theta_{ij}^R + \theta_{ij}^L + \theta_{ij}^T + \theta_{ij}^B - f_{ij} \geq 0 \end{cases} \quad (8)$$

$\theta_{ij}^d (d = R, L, T, B)$ are four binary variables to indicate whether room i is on the right, left, top or bottom side of room j. M is a constant large enough to ensure the inequalities in all situations. The last inequation indicates that either two rooms are not on the same floor ($f_{ij} = 0$), or they are on the same floor and should not overlap with each other ($f_{ij} = 0$ and at least

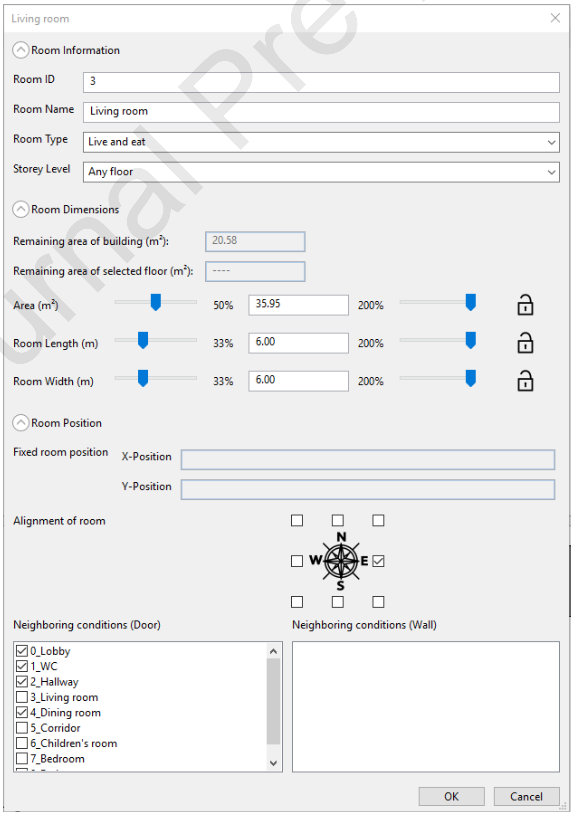

Figure 6: User interface of room dialog for editing requirements.

one θ_{ij} equals 0).

Boundary

A room can be placed on one of the edges of the floor's contour for specific need. For example (see figure 8 A), in northern hemisphere it is preferred to have living room or bedroom facing toward south for more sunshine. In this case, assuming there are n south edges on the floor k, the constraint is described in equation 9:

$$\begin{cases} x_i \geq x_{kl1} - M(1 - \beta_{kl} f_{ki}) \\ x_i + l_i \leq x_{kl2} + M(1 - \beta_{kl} f_{ki}) \\ y_i \leq y_{kl} + M(1 - \beta_{kl} f_{ki}) \\ \sum_{k=1,l=1} \beta_{kl} f_{ki} \geq 1 \end{cases} \quad (9)$$

β_{kl} is a binary variable denoting one of the south edges on floor k which is selected to be the adjacent edge to the room i. The first three inequations indicate that the bottom edge of the room and one of the south boundaries in the building overlap with each other. The last inequation ensures that such of a south boundary must exist in the building.

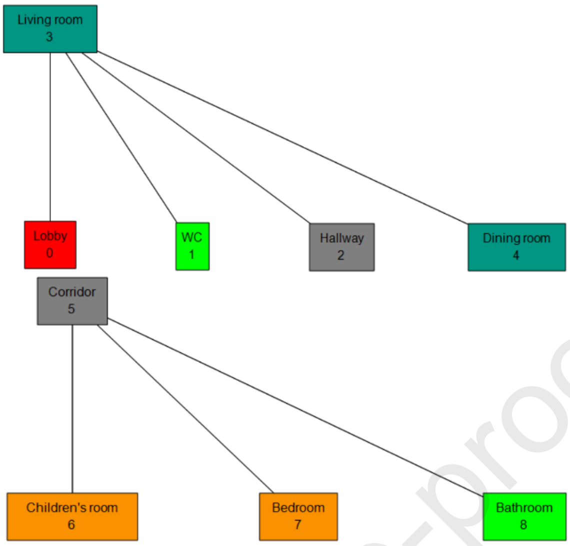

Figure 7: User interface of topological graph, representing adjacency of rooms.

Adjacency

The adjacency relationship between rooms can be specified for user's different preference. For example, a guest bathroom is usually adjacent to the entrance, or the master bathroom is connected to the master bedroom.

$$\begin{cases} x_i \le x_j + l_j - dc_{ij} \\ x_j \le x_i + l_i - dc_{ij} \\ y_i \le y_j + w_j - d(1 - c_{ij}) \\ y_j \le y_i + w_i - d(1 - c_{ij}) \\ \sum_{k=1} f_{ki} f_{kj} = 1 \end{cases} \tag{10}$$

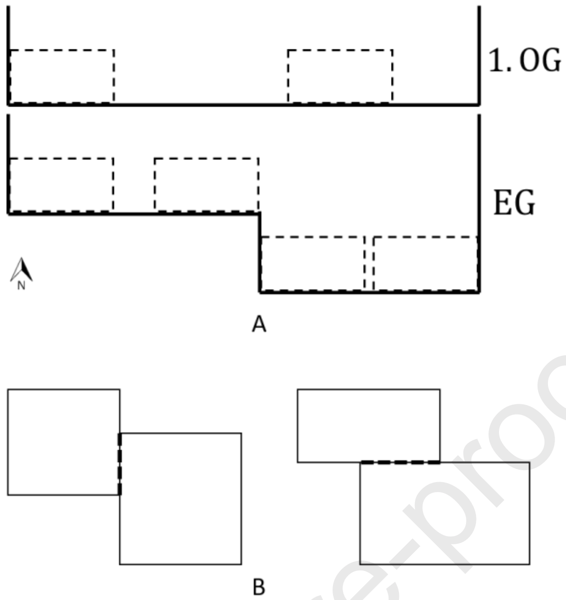

Figure 8: Example of requirements. A. Boundary requirement. B: Adjacency Requirement.

Two rooms can be either horizontally or vertically connected. When the binary variable $c_{ij} = 1$, it means two rooms are connected with an overlap of their top/bottom edges. In this case the first two inequations ensure a minimum connected length so that a door can be placed, and the third and fourth inequations forced such an overlap of their edges (see figure 8 B). The last inequation ensures that these two rooms have to be on the same floor.

There are requirements which depend on the room type or building type, and from such requirements specific constraints can be defined. For example, in a building with multiple floors, the position and dimension of the stairway on each floor should be the same, i.e. the value of (x_i, y_i, l_i, w_i). In addition, the space with type "Entrance" should be subject to a boundary constraint, which means it must be placed on one of the edge of the entrance floor's contour.

4.3. Solving mathematical optimization problem

The mathematical model which is formulated above is a mix-integer nonlinear programming (MINLP) problem, which can be solved by using mathematical solver based on the branch-and-cut algorithm [24] and interior point method [6]. In the SDaC Layout Designer, the SCIP Optimization Suite [3] is used. It consists of some software packages centered around the constraint integer programming framework SCIP, which are used to generate and solve mixed integer nonlinear programs. The SCIP framework provides various interfaces for different programming languages and mathematical modelling languages, while a number of solvers can also be linked and used by it.

The SCIP Optimization Suite allows full control of the solving process, hence it is possible to retrieve the intermediate results during the solving process. These results might not be globally optimal, but they are still feasible solutions and the building layouts derived from them could still be chosen for other reasons such as aesthetic aspect, since the layout design is by all means a user-driven task.

4.4. BIM model export

Both Streamer EDC and SDaC Layout Designer support export of openBIM model from the generated layouts. The results can be saved as IFC data, including the building elements such as floors, walls, doors and windows. The properties of these building elements are saved, as well as the connecting relationships of the walls and openings elements such as doors and windows.

Moreover, the space boundaries are also exported, providing the possibility of energy simulation in further use case.

5. Test Cases

Both projects have different use cases. Using a similar data set results in certain differences. The following list of rooms from a real-world example is used in both examples:

Room name	Area (m^2)
Lobby	4.42
WC	1.62
Hallway	3.84
Living room	35.95
Dining room	6.35
Corridor	3.84
Children's room	12.19
Bedroom	18.46
Bathroom	10.55

The outer hull of the building has a size of 6.2 m by 11.87 m and a wall thickness of 30 cm.

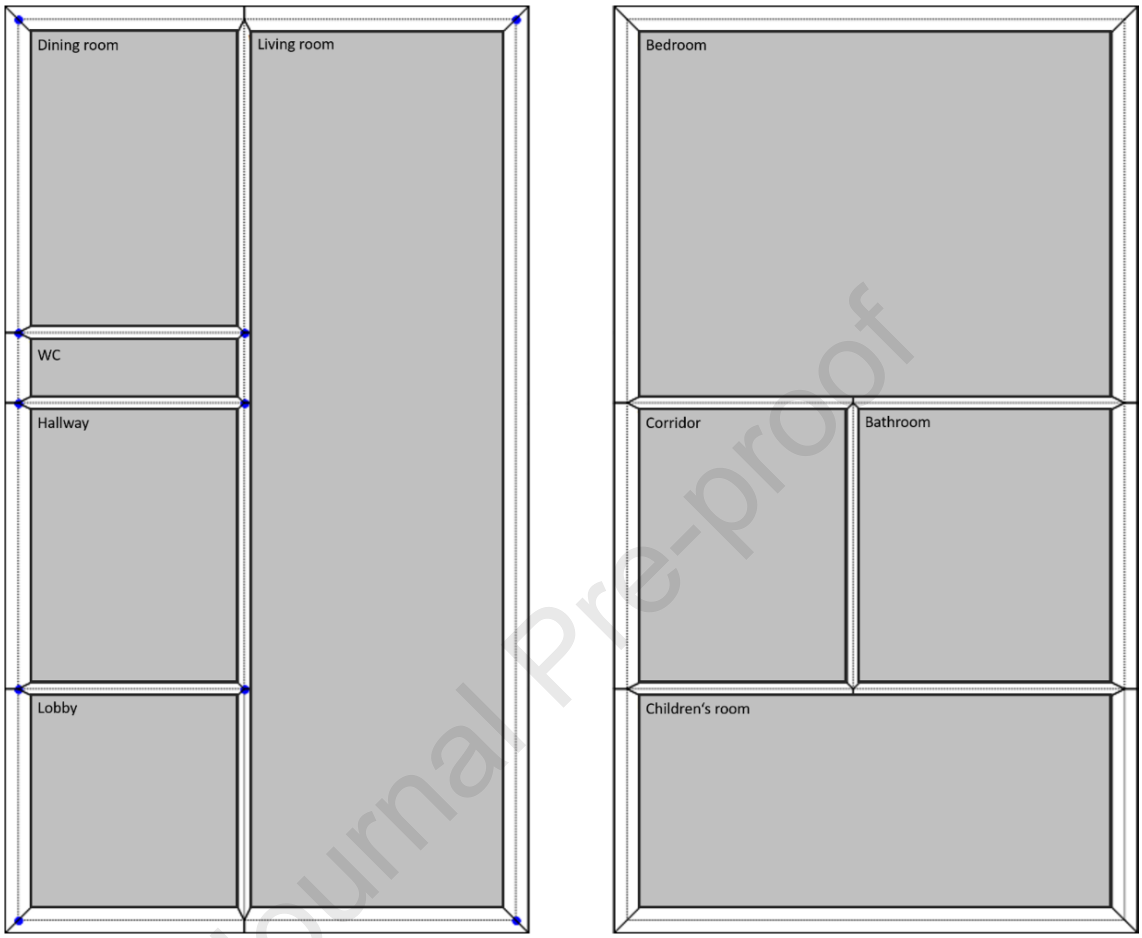

Figure 9: Example of a layout generated by EDC.

5.1. Examples of Streamer EDC

The room list has been extended with STREAMER specific default values (Room Type, Functional Area Type, Bouwcollege Layer, Hygienic Class, Access Security, User Profile, Equipment, Construction and Comfort Class). Figure 9 shows an example layout generated by EDC. Outer hull and free spaces have to be predefined. Corridors and stairs have been removed from the model, as they are set manually at a predefined position.

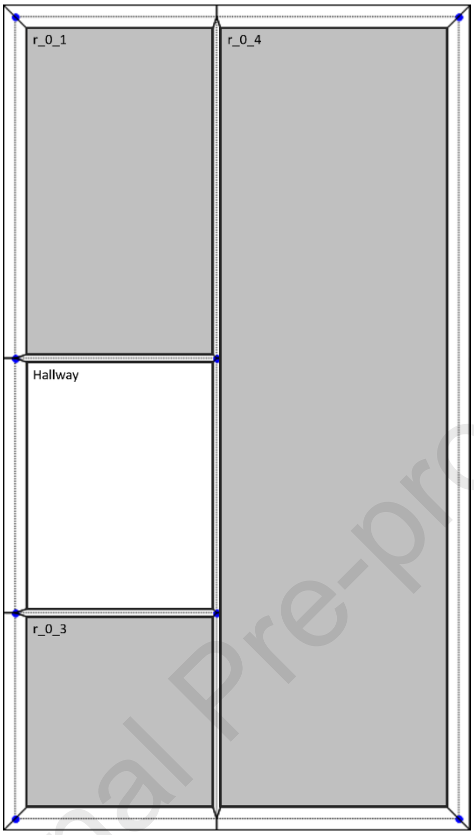

Figure 10: Limited predefined free spaces (gray) for a building, one story and a reserved area for stairs (white)

The Streamer EDC is not able to fix the position of a room automatically. The layout has to be manually pre-limited to a layout with restricted free spaces before the evolutionary computation (see figure 10).

5.2. Examples of SDaC Layout Designer

The room list above is added with more detail requirements such as target length, width, as well as some adjacency constraint. The extended room list is then applied in the SDaC Layout Designer. Figure 11 shows an example of layout generated by the SDaC Layout Designer, it represents the optimal solution of the mathematical programming problem. The walls are automatically added with a predefined parameter. The door between the rooms represents the adjacency constraint in the room list. A door is added for the room with type "Entrance" to connect to the outside.

Figure 12 shows another example of layout, which is retrieved from a suboptimal solution of the mathematical programming problem. Compared to those in figure 11, some of the rooms in this layout have different sizes and positions, which still fulfilling all the requirements in the room list.

5.3. Export of openBIM model

Figure 13 shows the IFC model exported from the generated result by SDaC Layout Designer, since it supports exporting more entities than Streamer EDC. The doors between rooms represent the adjacent constraint defined in the program of requirements. The door which connects the building and the outside is defined by the room type "Entrance". The IFC data can be read and edited by most CAD-Softwares such as Revit, ArchiCAD, etc.

6. Discussion and Comparison

In this paper, two approaches are proposed for the automatic layout generation, the Streamer EDC and the SDac Layout Designer. The Streamer EDC uses evolutionary strategies to generate layout for large-scale building. It combines the rules and program of requirements as input for the optimization process during the evolution. The SDaC Layout Designer relies more on a detailed program of requirements to generate layout for residential building.

6.1. Limit of introduced approaches

The building outline is restricted as axis-aligned polygon, while the rooms can only be rectangle. Although this is sufficient at the early design phase, but it still needs manual adjustment for the final design and construction. It is possible to use more complex rectilinear shapes in the optimization model, but this would increase its number of variables and constraints, resulting in much more time in the computation.

Figure 11: Example of optimal layout generated by SDaC Layout Designer

The Streamer EDC is developed for the layout generation of health-care building, which is mostly large scale. One of its most significant goals is to cluster rooms in different area to fulfill their functionalities. It is found out that the generated layouts differ most of the time, when the evolutionary computation has to be paused manually, and the theoretic optimum is difficult to reach in large-scale problem, meaning that the solution that is found is usually a random suboptimal solution.

The SDaC Layout Designer focuses more on proposing interior layouts for residential building, which means the requirements of every room are more specific with details. With basic requirements such as area limitation, various layouts can be retrieved from the suboptimal solutions found in the optimization. Since there are fewer constraints it would take much more time to search for all the feasible solutions. When more concrete requirements are added, such as boundary requirements and adjacency requirements, the feasible region, if still exists, is more limited and it would take less time for the solver to find feasible solutions.

One of the differences between the Streamer EDC and the SDaC Layout Designer is, how the room list is configured. In EDC the room type and the amount is given in every room definition, meaning there should be a certain number of rooms having the similar features. In the SDaC Layout Designer, every single room has its own definition. Besides, the rooms in Streamer EDC are connected through a corridor, which is a predefined structure that separates an area into smaller domains. Such structure would not be given any requirement like those of the rooms. In SDaC Layout Designer, however, such corridor should either be given as an individual "room" with specific adjacency requirement in the room list, or be predefined with fixed position and size.

6.2. Conclusion

As we see, both programs can generate interior layouts and export IFC data [15]. The core process in both programs is the comprehensive preparation of the program of requirement, which still requires certain human labor to configure. It is not easy to assure that from a given

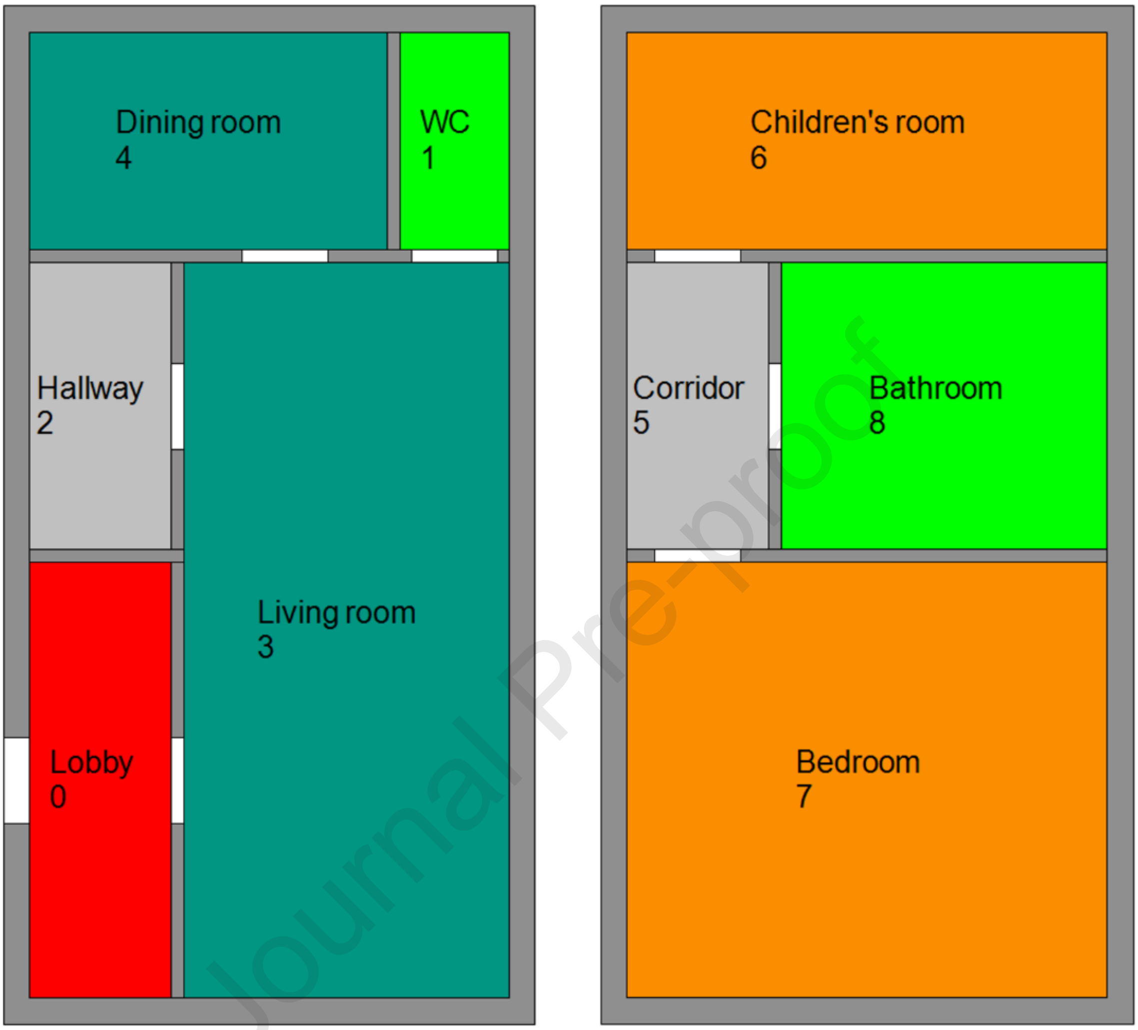

Figure 12: Example of suboptimal layout generated by SDaC Layout Designer

room list there is always a solution to represent a layout, therefore it would be more efficient when there is a system to verify if it is possible to generate a layout from the given room list. Such mechanism should be able to verify if the total usable area is large enough or not, or if the adjacency requirements can form a feasible topological graph. Moreover, generative artificial intelligence (GAI) is becoming more and more significant nowadays, if it could be applied to configure the room list, the generating process could be more efficient and more precise. For example, there is already research on generating topological graph by using neural networks [18]. But this requires a comprehensive analysis of present floorplans as training data, which would be a part of future research. Another possibility is to combine the clustering function in Streamer EDC and detail arrangement in SDaC Layout Designer. A hierarchical process could then be applied in different use cases such as large-scale buildings or residential house.

Acknowledgments

The work presented in this paper belongs to the EU funded STREAMER project and to the research project Smart Design and Construction (SDaC) funded by BMWK (Federal Ministry for Economic Affairs and Climate Action), funding ref. no. 01MN23005B.

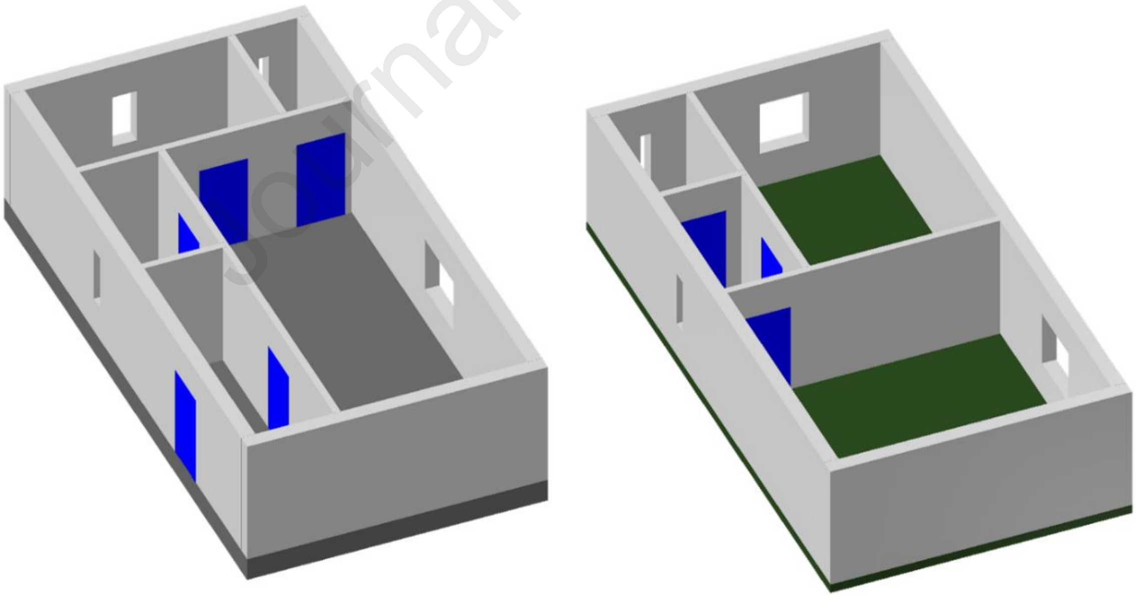

Figure 13: Exported 3D Model

References

[1] Abioye, S.O., Oyedele, L.O., Akanbi, L., Ajayi, A., Delgado, J.M.D., Bilal, M., Akinade, O.O., Ahmed, A., 2021. Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges. Journal of Building Engineering 44, 103299.

[2] Benner, J., Häfele, K.H., Bonsma, P., Bourdeau, M., Soubra, S., Sleiman, H., Robert, S., 2015. Interoperable tools for designing energy-efficient buildings in healthcare districts. CRC Press/Balkema.

[3] Bestuzheva, K., Besançon, M., Chen, W.K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., et al., 2021. The scip optimization suite 8.0. arXiv preprint arXiv:2112.08872 .

[4] Chaillou, S., 2020. Archigan: Artificial intelligence x architecture, in: Architectural intelligence. Springer, pp. 117–127.

[5] Culha, B., et al., 2005. An outside-in approach for product architecture and design, in: DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08. 2005, pp. 529–530

[6] Dikin, I., 1967. Iterative solution of problems of linear and quadratic programming, in: Doklady Akademii Nauk, Russian Academy of Sciences. pp. 747–748.

[7] Du, T., Turrin, M., Jansen, S., van den Dobbelsteen, A., Fang, J., 2020. Gaps and requirements for automatic generation of space layouts with optimised energy

performance. Automation in Construction 116, 103132. doi:https://doi.org/10.1016/j.autcon.2020.103132.

[8] Eastman, C., et al., 1974. An outline of the building description system. research report no. 50. .

[9] Eastman, C.M., 2018. Building product models: computer environments, supporting design and construction. CRC press.

[10] Egor, G., Sven, S., Martin, D., Reinhard, K., 2020. Computer-aided approach to public buildings floor plan generation. magnetizing floor plan generator. Procedia Manufacturing 44, 132–139.

[11] Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al., 1996. A density based algorithm for discovering clusters in large spatial databases with noise, in: kdd, pp. 226–231.

[12] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Advances in neural information processing systems 27.

[13] Hempel, S., Benner, J., Häefele, K.H., 2015. Generating early design alternatives based on formalized requirements and geospatial data, in: Proceedings of the CIB W.

[14] Herr, C.M., Ford, R.C., 2016. Cellular automata in architectural design: From generic systems to specific design tools. Automation in Construction 72, 39–45.

[15] ISO, 2013. 16739: 2013 industry foundation classes (ifc) for data sharing in the construction and facility management industries. International Organization for Standardization , 31.

[16] Koenig, R., Knecht, K., 2014. Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. AI EDAM 28, 285–299.

[17] Lopes, R., Tutenel, T., Smelik, R.M., De Kraker, K.J., Bidarra, R., 2010. A constrained growth method for procedural floor plan generation, in: Proc. 11th Int. Conf. Intell. Games Simul, Citeseer. Pp. 13-20.

[18] Madhawa, K., Ishiguro, K., Nakago, K., Abe, M., 2019. Graphnvp: An invertible flow model for generating molecular graphs. arXiv preprint arXiv:1905.11600 .

[19] Merrell, P., Schkufza, E., Koltun, V., 2010. Computer-generated residential building layouts, in: ACM SIGGRAPH Asia 2010 papers, pp. 1–12.

[20] Michalek, J., Choudhary, R., Papalambros, P., 2002. Architectural layout design optimization. Engineering optimization 34, 461–484.

[21] Nauata, N., Hosseini, S., Chang, K.H., Chu, H., Cheng, C.Y., Furukawa, Y., 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, pp. 13632–13641.

[22] van Nederpelt, S., Häfele, K.H., Hempel, S., Benner, J., Picinbono, G., Pols, J.P., 2015. Parametric modelling techniques for eeb. URL: http://www.streamer-project.eu/Downloads/D5.5STREAMER.pdf.

[23] Oprach, S., Bolduan, T., Steuer, D., Vössing, M., Haghsheno, S., 2019. Building the future of the construction industry through artificial intelligence and platform thinking. Digitale Welt 3, 40–44.

[24] Padberg, M., Rinaldi, G., 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM review 33, 60–100.

[25] Sebastian, R., Böhms, H., Bonsma, P., Van Den Helm, P., 2013. Semantic bim and gis modelling for energy-efficient buildings integrated in a healthcare district. ISPRS annals of the photogrammetry, remote sensing and spatial information sciences 2, 255–260.

[26] Shekhawat, K., Upasani, N., Bisht, S., Jain, R.N., 2021. A tool for computer-generated dimensioned floorplans based on given adjacencies. Automation in Construction 127, 103718.

[27] Ślusarczyk, G., 2018. Graph-based representation of design properties in creating building floorplans. Computer-Aided Design 95, 24–39.

[28] STREAMER-CONSORTIUM, 2013. Streamer - european research on energy-efficient healthcare districts. URL: http://www. streamer-project.eu/.

[29] Traversari, R., Den Hoed, M., Di Giulio, R., Bomhof, F., 2017. Towards sustainability through energy efficient buildings design: semantic labels. Entrepreneurship and Sustainability Issues 4, 243.

[30] Weise, M., Liebich, T., Nisbet, N., 2015. State-of-the-art review of advancements and challenges in ontology research. URL: https: //www.streamer-project.eu/Downloads/D5.1.pdf.

[31] Whitehead, B., Eldars, M., 1965. The planning of single-storey layouts. Building Science 1, 127–139.

[32] Wu, W., Fan, L., Liu, L., Wonka, P., 2018. Miqp-based layout design for building interiors, in: Computer Graphics Forum, Wiley Online Library. pp. 511–521.

[33] Zhong, Y., Geiger, A., 2023. Automated floorplan generation using mathematical optimization, in: ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022. CRC Press, pp. 404–410.

Highlights

Automated Building Layout Generation: Comparison of two Implemented Methods

Yingcong Zhong, Steffen Hempel, Andreas Geiger, Karl-Heinz Häfele, Veit Hagenmeyer

- Introduction of rule based floor plan generation from program of requirement (PoR) in early design phase.
- Evolutionary algorithms for large scale building layout generation.
- Mathematical programming for residential house layout generation.
- Generation of conceptional 3D building information models (BIM)

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Yingcong Zhong