Journal of Building Engineering 95 (2024) 110163

Contents lists available at ScienceDirect

Journal of Building Engineering

journal homepage: www.elsevier.com/locate/jobe

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

Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, Eggenstein-Leopoldshafen, 76344, Germany

A R T I C L E I N F O

Keywords:
Automated layout generation
BIM generation
Evolutionary strategies
Mathematical programming

A B S T R A C T

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 even the surrounding neighborhood's existing buildings, infrastructure, and natural environment. Particularly the requirements of interior layout design are usually coupled with each other, making the design process more complex and time-consuming. This paper proposes a novel manner to consider the requirements, aiming to facilitate the generation process of building's interior layout during the early design phase. Besides, this paper aims to generate vectored output, which is more applicable to export BIM model, which also benefits the subsequent design and construct steps. This paper introduces STREAMER Early Design Configurator (EDC) and SDaC Layout Designer, both of which formats the user's requirements, semi-automatically generates interior layouts and exports IFC models of the floor plans. STREAMER EDC is used to discover the potential layouts of large-scale buildings with many rooms such as hospitals, while SDaC Layout Designer focuses on residential houses with comparatively fewer rooms. 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.

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 [1], 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. Generative design frees the designers from the "trial and error" process, and with the application of artificial intelligence, it would also improve

* Corresponding author.
E-mail address: yingcong.zhong@kit.edu (Y. Zhong).

https://doi.org/10.1016/j.jobe.2024.110163
Received 22 November 2023; Received in revised form 2 July 2024; Accepted 6 July 2024
Available online 8 July 2024

productivity, safety and quality in not only for the layout designing, but also in other construction processes [2].

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 [3,4]. 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 BIM models become significantly important in the architect and construction industry.

In previous research of layout generation, either the user's requirements could not be considered, or the result of generation could not be of further usage. This paper presents novel workflows of interior layout generation in the early design phases, focusing on the formatting of the user's design requirements and automatic generation of openBIM data for further usage. According to different use cases, namely different scale and type of target building, the STREAMER EDC and the SDaC Layout Designer are presented. First of all a brief review of the state of the art introduces different methods or algorithms, which are 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 including how its mathematical model is built based on the program of requirements (PoR). As testing, STREAMER EDC is used to generate the layout of a hospital with three floors, while SDaC Layout Designer is used for the design of a residential house. The results from both tools have been 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 [5,6]. As is shown in Fig. 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 [7], 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 [8], 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.

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

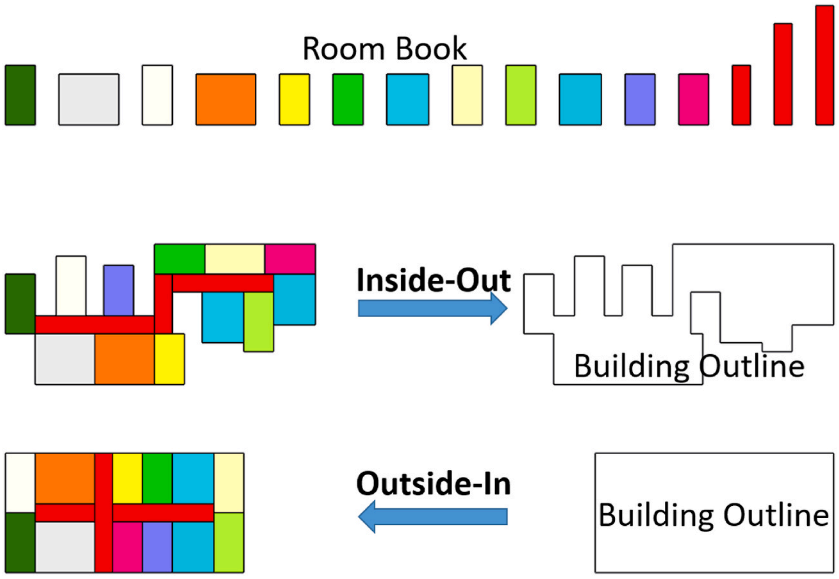


Fig. 1. Example of outside-in and inside-out design mode.

layouts for further usage. In Ref. [9] 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. Egor [10] presents a quasi-evolutionary strategy to generate floor plans in an iterative manner based on the connectivity and adjacency between specific rooms. Wang [11] introduces a method for the design of ship multi-deck compartment layout, which is similar to the building's interior layout design. A genetic algorithm is applied for the generation, in which the circulation intensity, adjacent intensity, penalty and ergonomic constraint is considered in the fitness function. Wu [12] introduces a mixed-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.

2.2. *Graph-theory aided*

There is research focusing on the adjacency between spaces in the layout design. This relationship is transformed into a topological graph, in which nodes represent spaces and edges represent connections. Such graph can be turned into a 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. In Ref. [13] Gan introduces a BIM-based graph data model to generate modular buildings automatically. The spatial attributes and the topological relationships, geometries and semantics are expressed with the help of IFC MVD. Shekhawat [14] presents a tool to construct dimensioned layouts using graph-theoretical and optimization techniques, in which the adjacency requirements are given either in an adjacency graph or in a dimensionless layout. Slusarczyk [15] presents hierarchical graph-based data structures for representing design solutions together with graph grammars, in which the local formulas expressing design properties are transformed into equivalent graph requirements. In Ref. [16] Wang presents an approach of dual graphs to generate floor plans. The dual graph of a generated floor plan either conforms to the dual graph of an input layout or is derived from the input's dual graph by applying transformation rules. With adjacency graph of rooms as input, rectangular floor plans are generated, which require further manual selection and adjustment to generate final design.

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. Herr [17] 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. By rasterizing the building footprint polygon into 2D grids, Lopes [18] proposes a room expansion algorithm to generate building layout that fulfills the adjacency and connectivity constraints. Sharafi [19] developed an approach for the automated spatial design of multi-story modular buildings, in which each design is presented by a three-dimensional design structure matrix. The architectural, structural, constructional and environmental objectives are modeled as constraints in the approach.

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. Chaillou [20] trains generative adversarial networks (GAN) [21] with real cases floorplan pictures to generate building footprints, split rooms and place furniture step by step. Merrell [22] presents a Bayesian Network to structurally learn the adjacency of rooms, from which the results are further optimized for detailed design. Nauata [23] also trains GANs and uses topological graph diagram as input for geometric design of floorplans. In Ref. [24] a hybrid approach is presented, in which agent-based modeling and deep learning method are combined for the generation with a building's footprint as input. In the agent-based modeling the topological relationships of spaces are generated and is used as the input for a cGAN (conditional GAN) model, which then generated the 2D layout. Tanasra [25] introduces a GAN model for furnishing interior space planning. Wang [26] combines deep learning and graph algorithms for the layout generation. Wu [27] uses data-driven method for the interior layout generation. Zhao [28] presents a method of GAN, which could be applied in both synthesis and analysis of layout plans of emergency departments in general hospitals.

2.5. *Overview*

One of the challenges of layout generation is, the input from the user is inevitably required, otherwise it will be time consuming to pick up preferred results from thousands of random computer-generated solutions. In addition, the results are expected to be vectored data, so that they can be reused in other applications for further validation and simulation. The aforementioned methods have shown their own advantages in the layout generation process, but still have certain limitation. With mathematical programming, it is easy to generate vectored result for openBIM export, but it requires a comprehensive description of user's requirements to represent the constraints. The layout generated by cell-assignment method is usually restricted within a predefined boundary, which limits the generation of possible variance. With the graph-theory aided method it is easy to fulfill topological requirements and generate more topological variances, but it also requires further processes to finish the geometric design. Machine learning has proven its strength in generative design with variances, but it requires abundant training data before generation and the result also needs further vectorization. Compared to aforementioned researches, the methods in this paper show the novelty of fulfilling the user's requirements for early design, especially the clustering and adjacency of specified spaces, not only those on the same floor, but also those on different floors. In addition, an automatic BIM generation process is integrated into both STREAMER EDC and SDaC Layout Designer, which enables integration into a BIM workflow, as both systems generate vectored data. Although the generated layouts fulfill the basic requirements, they are still conceptual design and need aesthetic adjustment by architects (Computer Aided Architectural Design, CAAD). BIM model (IFC data) also makes thermal simulation and life-cycle assessment possible in the further construction processes. With the help of BIM models, these design and construction processes become successive and barrier-free.

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 [29,30,31]. Goal of the STREAMER project was to increase energy efficiency of health-care buildings [32]. 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
- Project specific requirements

These parameters are defined in a STREAMER-owned common rule set language [33].

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 …) [34,35].

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

After loading the PoR and the rule set [33], the algorithm is started and begins with an empty floor plan as shown in Fig. 3.

The initial state is further processed in parallel. A certain number of mutations is applied 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 by 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

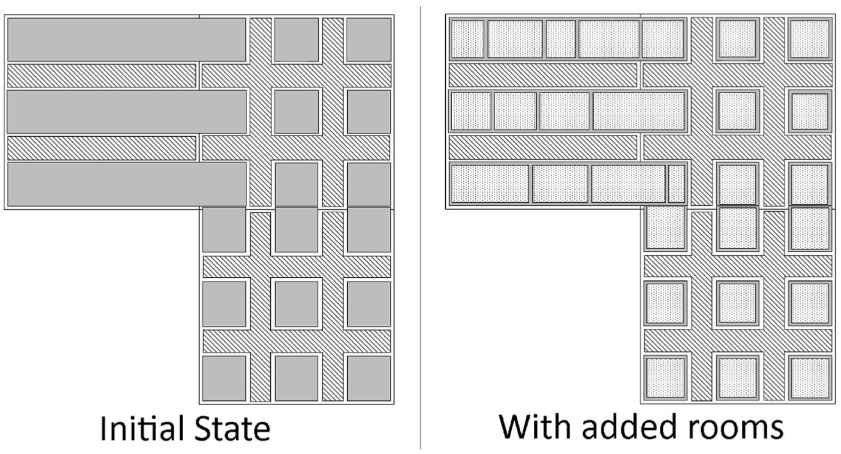


Fig. 2. Example of corridor mode in the EDC, corridor layout hatched, resulting free space in grey, initial state without rooms left, added rooms right.

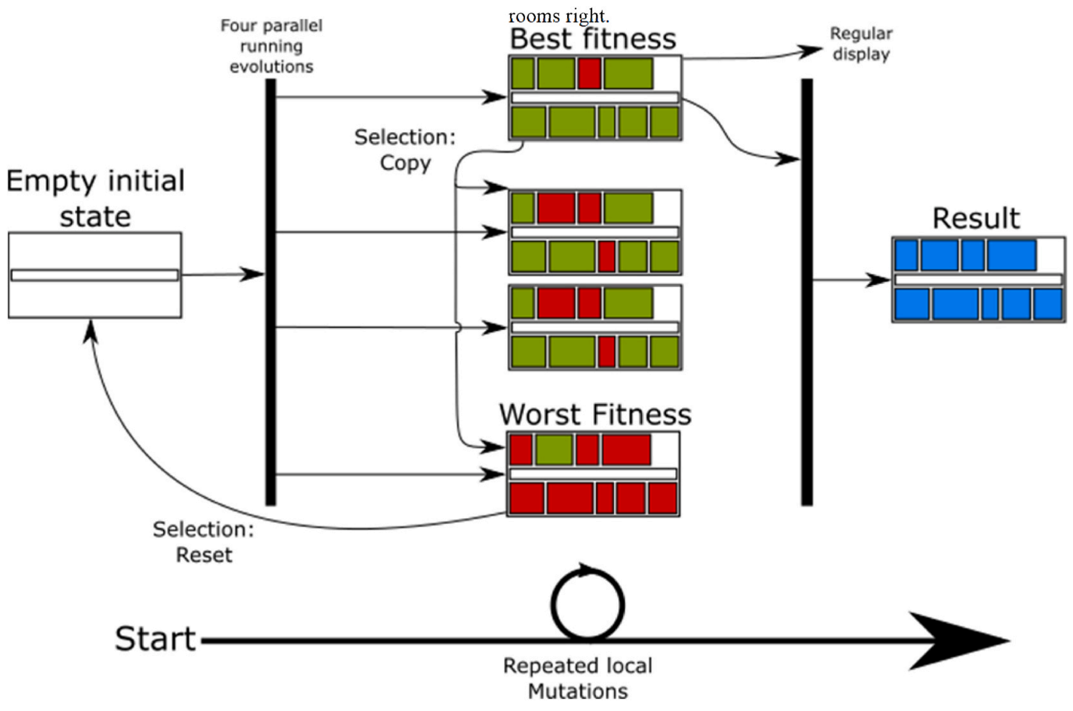


Fig. 3. Streamer Layout generation process.

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 one or two sets and apply a relation to them with predefined parameters. Rules using a single set apply a relationship between all rooms in that set. Rules using two sets apply their relation between rooms from one set to the rooms in the other set.

Parameters assign values required to describe relations, e.g. the relation *maximum distance* has a parameter that describes the value of the maximum distance in meters.

Relations applying to one set are:

(1) Cluster: All rooms must be connected.
(2) Cluster (same floor): All rooms must be on the same floor and connected.
(3) Distance to outer wall: All rooms must be at a minimum or maximum distance to the outer wall.
(4) Predefined floor: All rooms must be at the same floor.
(5) Relations applying to two sets are:
(6) Minimum/maximum distance: All rooms from a set must be at a minimum/maximum distance from another set.
(7) Same/different floor: All rooms from a set must be on the same/a different floor than rooms from another set.
(8) (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):

$$\mathrm{fitness}_{\mathrm{global}}(\mathrm{V}) = \sum_{\mathrm{i}=0}^{\text{number of rules}} \mathrm{fitness}(\mathrm{V}_{\mathrm{i}}) \cdot \mathrm{priority}_{\mathrm{i}} \tag{1}$$

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

$$\text{fitness}(v) = \begin{cases} \text{fitness}_{\text{soft}}(v) \\ \text{fitness}_{\text{hard}}(v) \\ \text{fitness}_{\text{combined}}(v) \end{cases} \tag{2}$$

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

$$\text{fitness}_{\text{soft}}(v) = \text{how good the rule is fulfilled} \tag{3}$$

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. The actual values used for constants (worst/best fitness) and as results from the fitness function are depending on the implementation.

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

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.

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

3.2.2. Rule implementation for distance

Example: 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. Rule implementation for cluster

Example: 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 rule needs no parameters.

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

DBSCAN [36] 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 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 Fig. 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 [37]. 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 [38]. 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 CAD software.

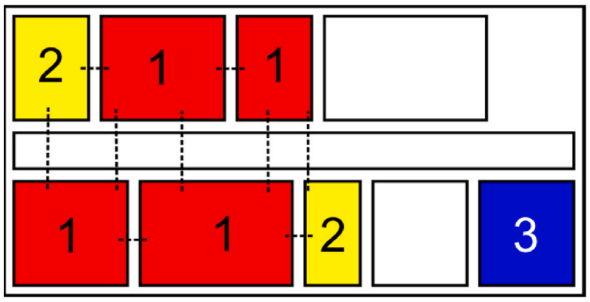


Fig. 4. DBSCAN applied to rooms, red: core points. yellow: reachable points. blue: noise.

4.1. Workflow of the program

Fig. 5 presents the workflow of SDaC Layout Designer. 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 are 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 converted 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 to an 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.

4.1.2. Definition of requirements

The program of requirements can either be imported from CSV data or directly created in the program. The user can define the dimension of a room, as well as its neighboring relationship with the other rooms. Optionally the position of certain room can also be fixed to a certain point or on a certain direction. Fig. 6 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):

$$\sum_{k=1}^{K} f_{ik} = 1 \tag{6}$$

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):

$$\min \sum_{1}^{K} A(k) - \sum_{1}^{N} l_i w_i + \sum_{1}^{N} (l_i - l_{ti})^2 + (w_i - w_{ti})^2 \tag{6}$$

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.

4.2.1. Non-overlap

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

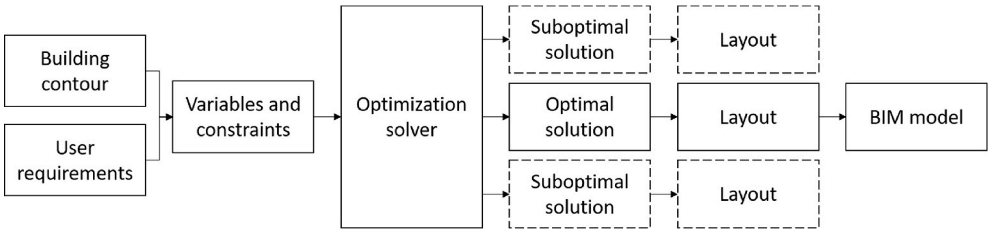


Fig. 5. SDaC Layout Designer workflow.

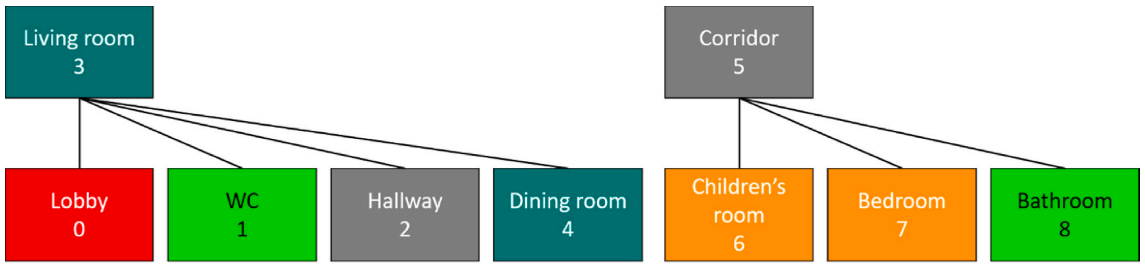


Fig. 6. User interface of topological graph, representing adjacency of rooms.

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

$\theta_{ij}^d(d = R, L, T, B)$ 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. $f_{ij} = \sum_k^K \left(f_{ki} f_{kj}\right)$ is an auxiliary variable to determine if two rooms are on the same floor or not. The last inequation indicates that either two rooms are not on the same floor ($f_{ij} = 0$), so that their projective overlap doesn't matter, or they are on the same floor and should not overlap with each other ($f_{ij} = 1$ and at least one θ_{ij} equals 1).

4.2.2. Boundary

A room can be placed on one of the edges of the floor's contour for specific need. For example (see Fig. 7 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):

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

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

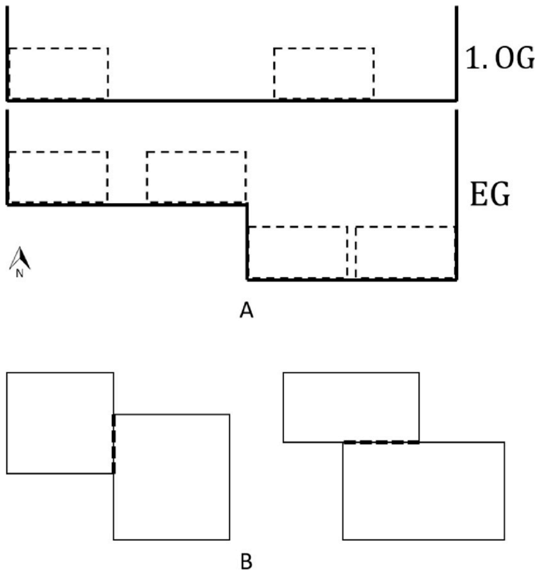


Fig. 7. Example of requirements. A. Boundary requirement. B: Adjacency Requirement.

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.

4.2.3. 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.

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

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 Fig. 7 B). The last inequation ensures that these two rooms have to be on the same floor (equation (10)).

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 mixed-integer nonlinear programming (MINLP) problem, which can be solved by using mathematical solver based on the branch-and-cut algorithm [39] and interior point method [40]. In the SDaC Layout Designer, the SCIP Optimization Suite [41] is used. It consists of 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 stored, as well as the connecting relationships of the walls and openings elements such as doors and windows. In addition, the space boundaries are also exported, providing the possibility of energy simulation in further use case.

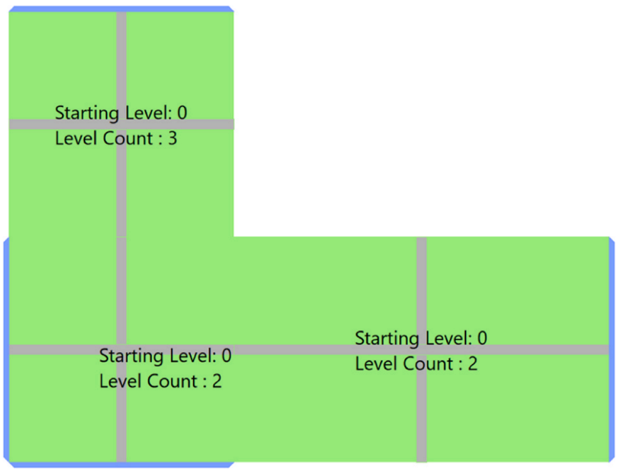


Fig. 8. STREAMER example, building boundary.

5. Test cases

STREAMER EDC and SDaC Layout Designer are used in different use cases. STREAMER EDC is used for the layout generation of hospitals and SDaC Layout Designer for residential house.

5.1. *Examples of STREAMER EDC*

Fig. 8 shows the building's predefined boundary and the program of requirements. Green area is buildable space in which rooms are placed, while the grey area is a given corridor pattern. Fig. 9 shows the program of requirements. 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). Applied rules are:

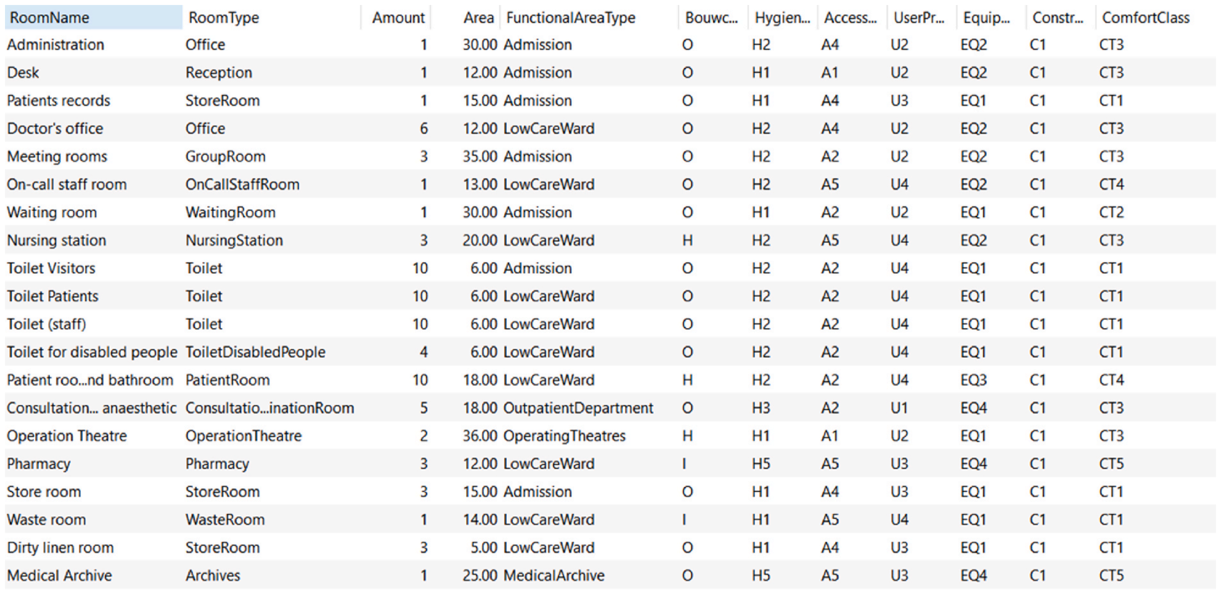
(1) Rooms with functional area type "Admission" must be on the same floor
(2) Rooms with functional area type "MedicalArchive" must be on the same floor
(3) Rooms with functional area type "LowCareWard" must be clustered
(4) Rooms with functional area type "PatientRoom" must be at minimum distance to rooms with functional area type "NursingStation"
(5) Rooms with hygienic class "H5" must be clustered

Fig. 10 demonstrates how the genetic algorithms performs in the layout generation. Each iteration represents a cycle as shown in the middle part of Fig. 3. As more iterations are executed, the fitness value gets closer to the theoretical optimum. Each run contains 300 thousands iterations and 10 runs are repeated with the same starting condition (same starting fitness). It can be seen that as the evolution is random, there is a gap between the best and worst fitness value. Additionally, runs might reach a local minimum, and running more iterations can leave this minimum and reach better results. Fig. 11 shows the generated 2D layout, in which different color represents different functional area type as shown in Fig. 9.

5.2. *Examples of SDaC layout designer*

Fig. 12 presents the program of requirements used in SDaC Layout Designer for the generation of residential house witha graph display the adjacency between rooms. The two-floor building's contour is a rectangle with 6.2 m * 11.87 m. Fig. 13 presents the progress of optimization, in which the x-axis represents the step when a suboptimal solution is found or the global optimum is reached, and the y-axis is the optimized goal value. 10 runs are shown in this line chart to show how the optimization performs. It is important to note that the x-axis doesn't represent the linear time interval, nor is comparable to the iteration in the STREAMER EDC, but denotes the step when a feasible solution is found. The optimization starts with random initial values, and the optimized values of suboptimal solutions differ randomly in each run, but it reaches the same global optimum eventually in every run. Fig. 14 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.

It is possible to retrieve a suboptimal solution of the mathematical programming problem and generate another layout from it. Compared to the rooms in Fig. 14, some of the rooms in this layout have different sizes and positions, which are still fulfilling all the



RoomName	RoomType	Amount	Area	FunctionalAreaType	Bouwc...	Hygien...	Access...	UserPr...	Equip...	Constr...	ComfortClass
Administration	Office	1	30.00	Admission	O	H2	A4	U2	EQ2	C1	CT3
Desk	Reception	1	12.00	Admission	O	H1	A1	U2	EQ2	C1	CT3
Patients records	StoreRoom	1	15.00	Admission	O	H1	A4	U3	EQ1	C1	CT1
Doctor's office	Office	6	12.00	LowCareWard	O	H2	A4	U2	EQ2	C1	CT3
Meeting rooms	GroupRoom	3	35.00	Admission	O	H2	A2	U2	EQ2	C1	CT3
On-call staff room	OnCallStaffRoom	1	13.00	LowCareWard	O	H2	A5	U4	EQ2	C1	CT4
Waiting room	WaitingRoom	1	30.00	Admission	O	H1	A2	U2	EQ1	C1	CT2
Nursing station	NursingStation	3	20.00	LowCareWard	H	H2	A5	U4	EQ2	C1	CT3
Toilet Visitors	Toilet	10	6.00	Admission	O	H2	A2	U4	EQ1	C1	CT1
Toilet Patients	Toilet	10	6.00	LowCareWard	O	H2	A2	U4	EQ1	C1	CT1
Toilet (staff)	Toilet	10	6.00	LowCareWard	O	H2	A2	U4	EQ1	C1	CT1
Toilet for disabled people	ToiletDisabledPeople	4	6.00	LowCareWard	O	H2	A2	U4	EQ1	C1	CT1
Patient roo...nd bathroom	PatientRoom	10	18.00	LowCareWard	H	H2	A2	U4	EQ3	C1	CT4
Consultation... anaesthetic	Consultatio...inationRoom	5	18.00	OutpatientDepartment	O	H3	A2	U1	EQ4	C1	CT3
Operation Theatre	OperationTheatre	2	36.00	OperatingTheatres	H	H1	A1	U2	EQ1	C1	CT3
Pharmacy	Pharmacy	3	12.00	LowCareWard	I	H5	A5	U3	EQ4	C1	CT5
Store room	StoreRoom	3	15.00	Admission	O	H1	A4	U3	EQ1	C1	CT1
Waste room	WasteRoom	1	14.00	LowCareWard	I	H1	A5	U4	EQ1	C1	CT1
Dirty linen room	StoreRoom	3	5.00	LowCareWard	O	H1	A4	U3	EQ1	C1	CT1
Medical Archive	Archives	1	25.00	MedicalArchive	O	H5	A5	U3	EQ4	C1	CT5

Fig. 9. STREAMER example, program of requirements.

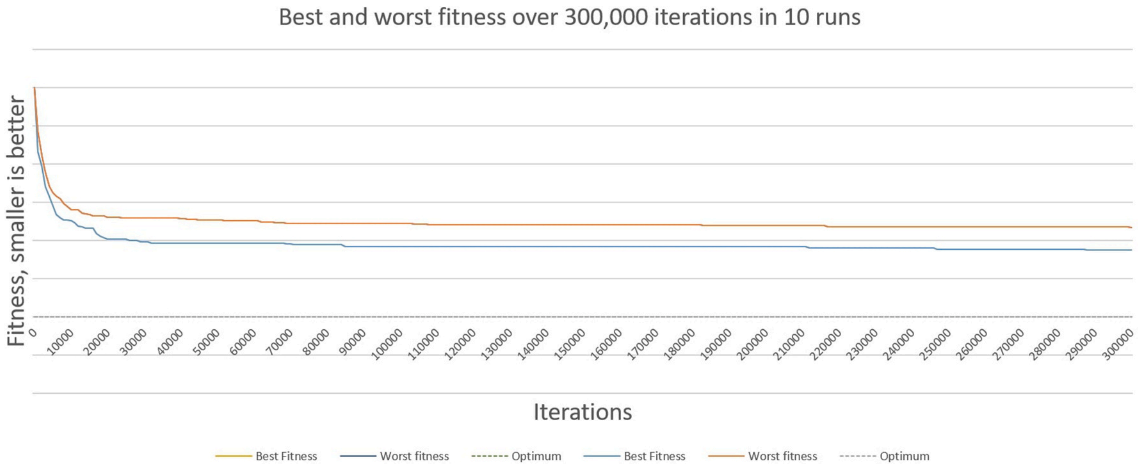


Fig. 10. STREAMER example, progress of generation.

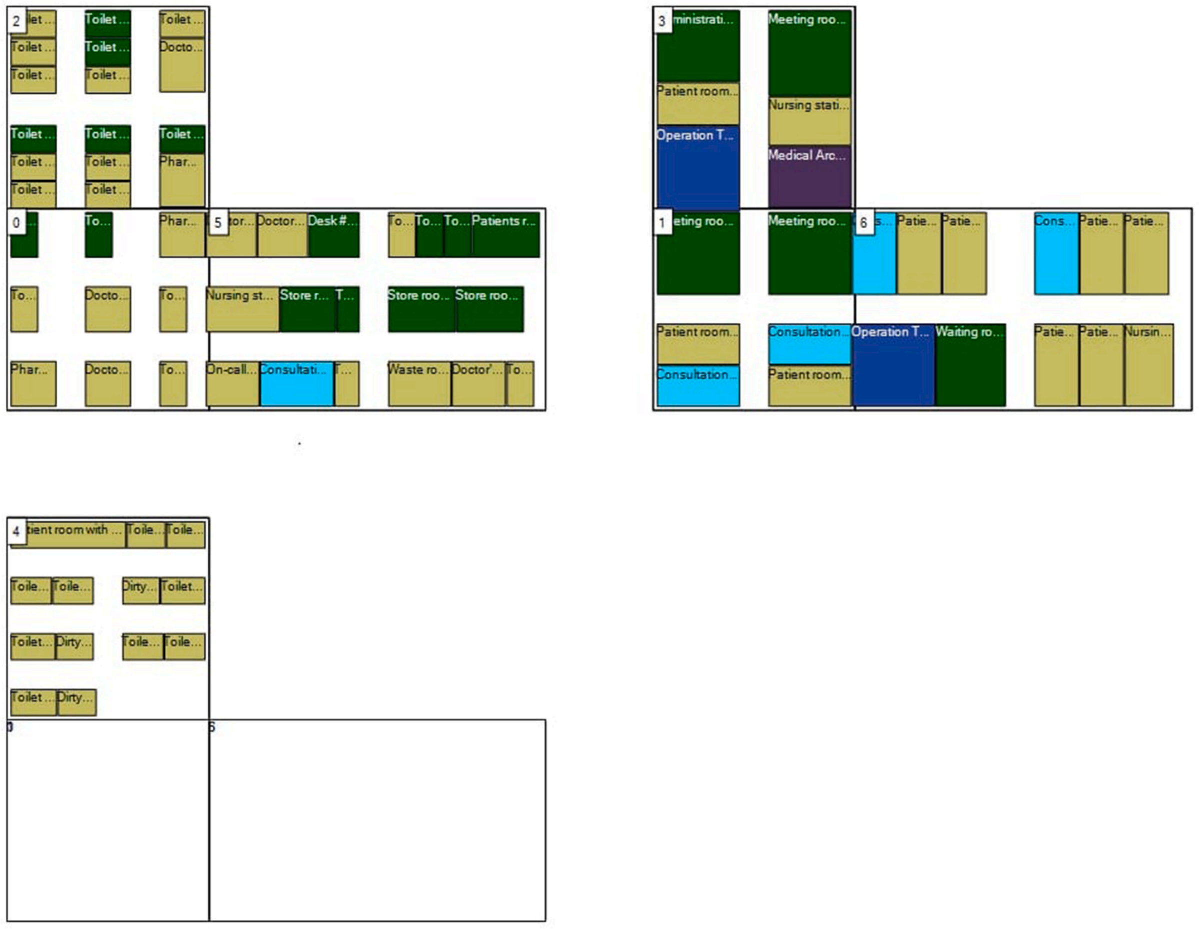
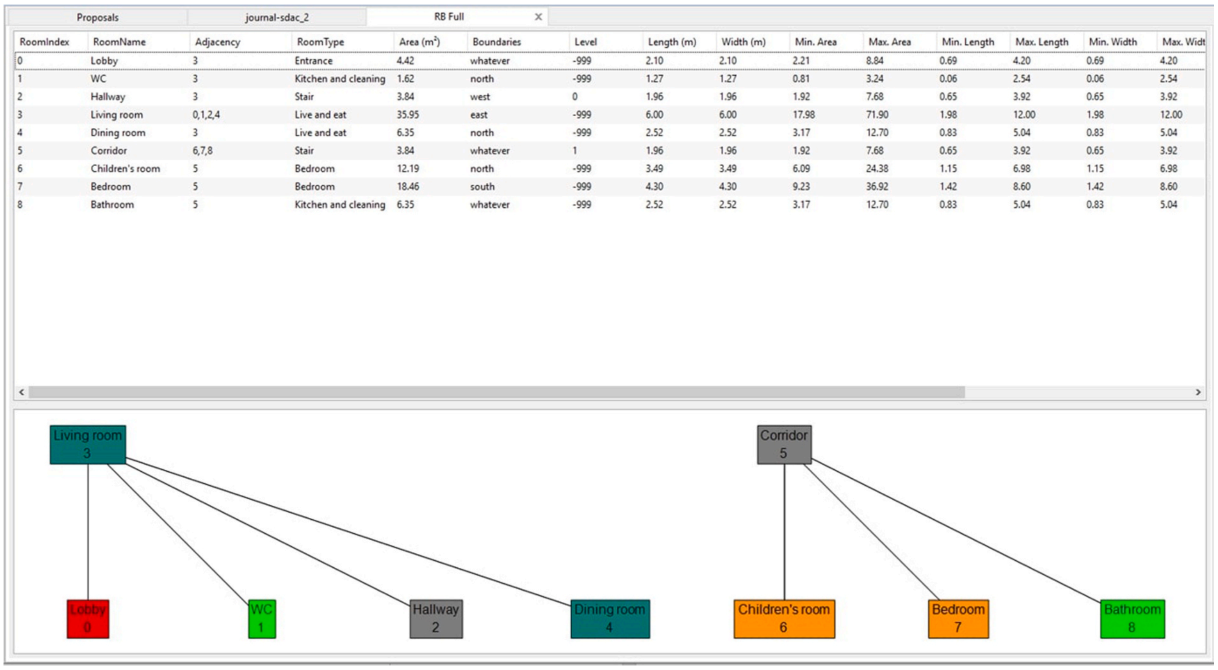


Fig. 11. STREAMER example, 2D layout (screenshot from the software).

requirements in the room list.

5.3. *Export of openBIM model*

Figs. 15 and 16 show the 3D IFC models exported from the generated results by SDaC Layout Designer and by STREAMER EDC. To



RoomIndex	RoomName	Adjacency	RoomType	Area (m^2)	Boundaries	Level	Length (m)	Width (m)	Min. Area	Max. Area	Min. Length	Max. Length	Min. Width	Max. Widt
0	Lobby	3	Entrance	4.42	whatever	-999	2.10	2.10	2.21	8.84	0.69	4.20	0.69	4.20
1	WC	3	Kitchen and cleaning	1.62	north	-999	1.27	1.27	0.81	3.24	0.06	2.54	0.06	2.54
2	Hallway	3	Stair	3.84	west	0	1.96	1.96	1.92	7.68	0.65	3.92	0.65	3.92
3	Living room	0,1,2,4	Live and eat	35.95	east	-999	6.00	6.00	17.98	71.90	1.98	12.00	1.98	12.00
4	Dining room	3	Live and eat	6.35	north	-999	2.52	2.52	3.17	12.70	0.83	5.04	0.83	5.04
5	Corridor	6,7,8	Stair	3.84	whatever	1	1.96	1.96	1.92	7.68	0.65	3.92	0.65	3.92
6	Children's room	5	Bedroom	12.19	north	-999	3.49	3.49	6.09	24.38	1.15	6.98	1.15	6.98
7	Bedroom	5	Bedroom	18.46	south	-999	4.30	4.30	9.23	36.92	1.42	8.60	1.42	8.60
8	Bathroom	5	Kitchen and cleaning	6.35	whatever	-999	2.52	2.52	3.17	12.70	0.83	5.04	0.83	5.04

Fig. 12. SDaC example, program of requirements.

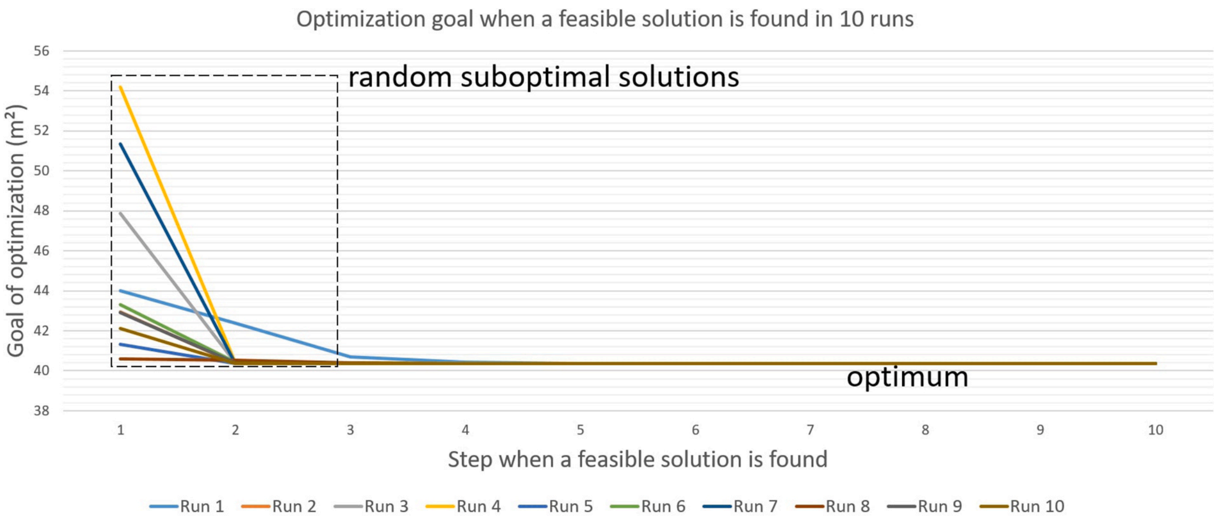


Fig. 13. SDaC example, progress of optimization.

demonstrate the usability of the generated IFC models, the models were imported and edited in the BIM authoring applications REVIT and ArchiCAD.

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. Both can transform the user's requirements into machine-readable format and generate interior layouts that can be exported as IFC format for further usage.

Another difference 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 similar features. In the SDaC Layout Designer, every single room has its own definition. Besides, the rooms in.

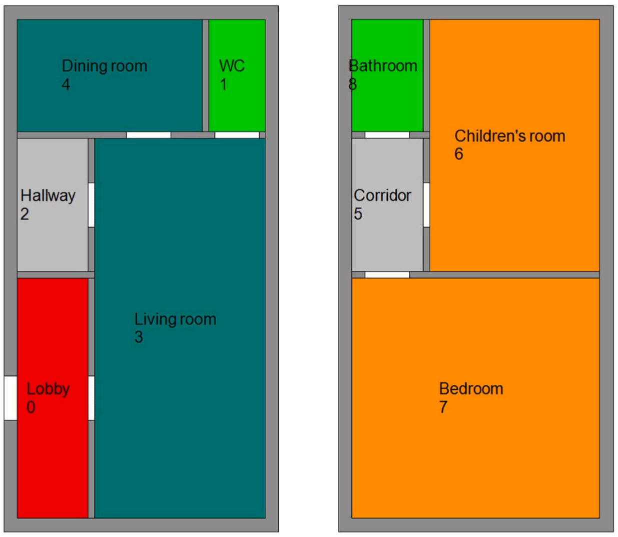


Fig. 14. Example of optimal layout generated by SDaC Layout Designer.

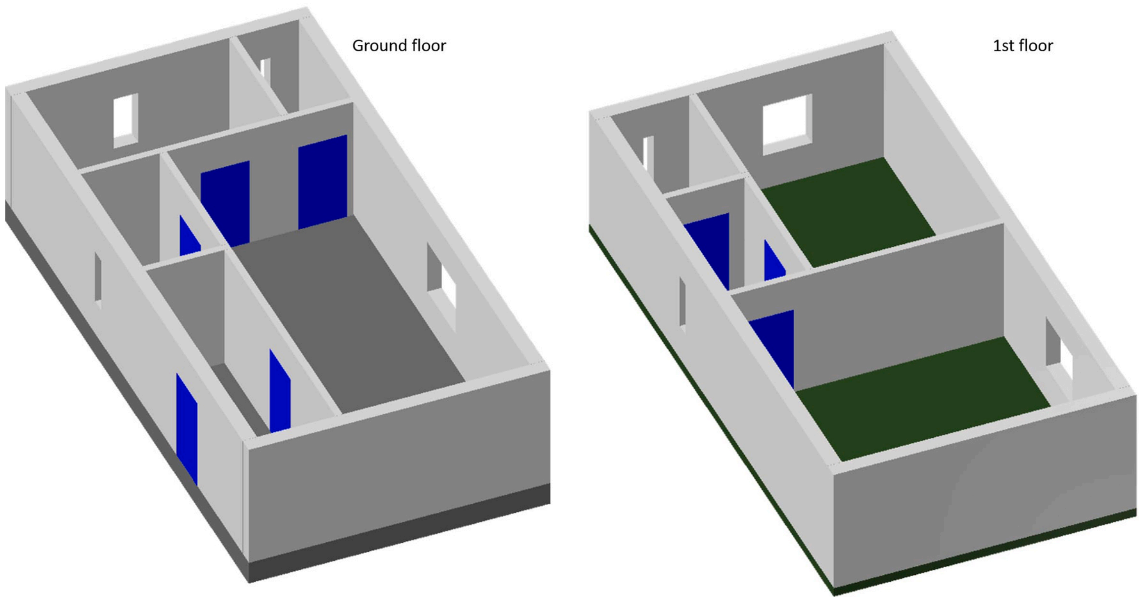


Fig. 15. SDaC example, exported IFC model.

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 corridors should either be given as an individual "room" with specific adjacency requirement in the room list, or be predefined with fixed position and size. Further comparison is listed in Table 1.

6.1. Limit of introduced approaches

In both STREAMER EDC and SDaC Layout Designer, the building outline is restricted as axis-aligned polygon, while the rooms of both methods can only be rectangular. Although this is sufficient at the early design phase, it still needs manual adjustment for the final design and construction. Theoretically 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 longer computation time. The effective generation of more complex geometries would be a part of the future research.

The STREAMER EDC is developed for the layout generation of large-scale health-care buildings. One of its most significant goals is to cluster rooms with functional relations. The theoretic optimum is difficult to reach in large-scale problem, meaning that the solution that is found is usually a random suboptimal solution.

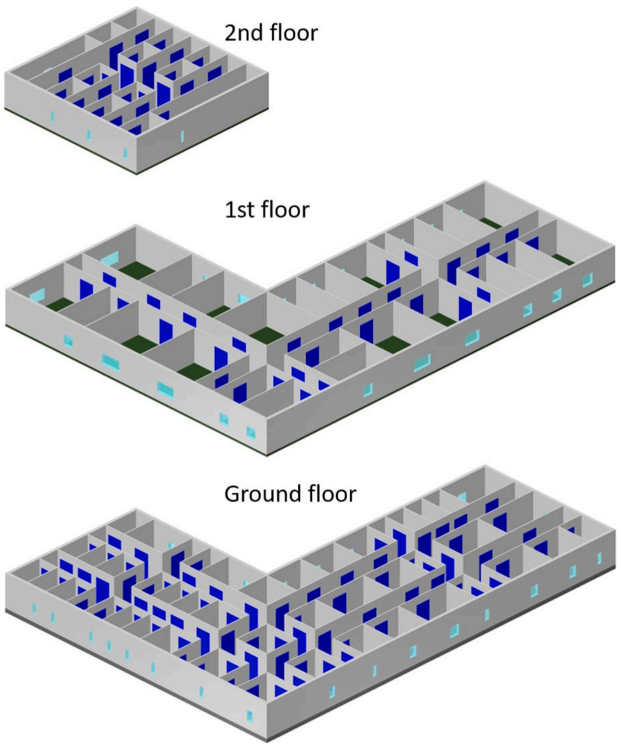


Fig. 16. STREAMER example, exported IFC Model.

Table 1
Comparison between STREAMER EDC and SDaC layout designer.

	STREAMER EDC	SDaC Layout Designer
Runtime	Infinite, until cancellation by user	Theoretically finite
Traceability	Pseudo-random, local maximum reproducible within same time and with same seed	Global optimum reproducible
Room placement	Insertion of rooms only in free space	Insertion of rooms freely within boundary
Rule definition	Any rule, as long as it could be mapped onto a fitness value	Any rule, as long as it can be a weighted optimization goal
Adjacency	Described as rule, a mandatory constraint	Described as constraints and can be represented in graph
Corridor	Set by layout template or fixed	Defined as a room in PoR
Access to building	Only set by rule, ground floor is defined in building boundary	Defined as a room in PoR
Connection between floors	Set by rule above/below; reserved space in predefined layout	Staircase is defined in PoR

The SDaC Layout Designer focuses more on proposing interior layouts for residential building, which means the requirements of every room are more specific. 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, is more limited and it takes less time for the solver to find feasible solutions. However, this would take more time for the preparation of PoR.

6.2. Conclusion

As is shown, both STREAMER EDC and SDaC Layout Designer can generate interior layouts and export IFC data [42]. The core process in both programs is the comprehensive preparation of the program of requirement as input, which still requires certain human labor to configure. It is not easy to assure that from a given 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 useable 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 [43]. 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, such that a hierarchical process could then be applied in different use cases such as large-scale buildings or residential houses.

Complement

The SDaC Layout Designer is provided as Freeware under the link: https://www.iai.kit.edu/1302.php.

CRediT authorship contribution statement

Yingcong Zhong: Writing – review & editing, Writing – original draft, Software, Methodology. **Steffen Hempel:** Writing – review & editing, Visualization, Validation, Software, Conceptualization. **Andreas Geiger:** Writing – review & editing, Validation. **Karl-Heinz Haefele:** Writing – review & editing. **Veit Hagenmeyer:** Writing – review & editing.

Declaration of competing interest

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.

Data availability

Data will be made available on request.

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.

References

[1] B. Whitehead, M. Eldars, The planning of single-storey layouts, Build. Sci. 1 (1965) 127–139.
[2] S.O. Abioye, L.O. Oyedele, L. Akanbi, A. Ajayi, J.M.D. Delgado, M. Bilal, O.O. Akinade, A. Ahmed, Artificial intelligence in the construction industry: a review of present status, opportunities and future challenges, J. Build. Eng. 44 (2021) 103299.
[3] C. Eastman, et al., An Outline of the Building Description System, 1974 research report no. 50.
[4] C.M. Eastman, Building Product Models: Computer Environments, Supporting Design and Construction, CRC press, 2018.
[5] B. Culha, et al., 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, 2005, pp. 529–530.
[6] R. Weber, C. Mueller, C. Reinhart, Automated floorplan generation in architectural design: a review of methods and applications, Autom. ConStruct. 140 (2022) 104385.
[7] R. Koenig, K. Knecht, Comparing two evolutionary algorithm based methods for layout generation: dense packing versus subdivision, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 28 (2014) 285–299.
[8] T. Du, M. Turrin, S. Jansen, A. van den Dobbelsteen, J. Fang, Gaps and requirements for automatic generation of space layouts with optimised energy performance, Autom. ConStruct. 116 (2020) 103132, https://doi.org/10.1016/j.autcon.2020.103132.
[9] J. Michalek, R. Choudhary, P. Papalambros, Architectural layout design optimization, Eng. Optim. 34 (2002) 461–484.
[10] G. Egor, S. Sven, D. Martin, K. Reinhard, Computer-aided approach to public buildings floor plan generation. magnetizing floor plan generator, Procedia Manuf. 44 (2020) 132–139.
[11] Yun-long Wang, Zhang-pan Wu, Guan Guan, Kai Li, Shu-hong Chai, Research on intelligent design method of ship multi-deck compartment layout based on improved taboo search genetic algorithm, Ocean Eng. 225 (2021) 108823.
[12] W. Wu, L. Fan, L. Liu, P. Wonka, Miqp-based layout design for building interiors, in: Computer Graphics Forum, Wiley Online Library, 2018, pp. 511–521.
[13] Vincent JL. Gan, BIM-based graph data model for automatic generative design of modular buildings, Autom. ConStruct. 134 (2022) 104062.
[14] K. Shekhawat, N. Upasani, S. Bisht, R.N. Jain, A tool for computer-generated dimensioned floorplans based on given adjacencies, Autom. ConStruct. 127 (2021) 103718.
[15] G. Ślusarczyk, Graph-based representation of design properties in creating building floorplans, Comput. Aided Des. 95 (2018) 24–39.
[16] X. Wang, Y. Yang, K. Zhang, Customization and generation of floor plans based on graph transformations, Autom. ConStruct. 94 (2018) 405–416.
[17] C.M. Herr, R.C. Ford, Cellular automata in architectural design: from generic systems to specific design tools, Autom. ConStruct. 72 (2016) 39–45.
[18] R. Lopes, T. Tutenel, R.M. Smelik, K.J. De Kraker, R. Bidarra, A constrained growth method for procedural floor plan generation, in: Proc. 11th Int. Conf. Intell. Games Simul, Citeseer, 2010, pp. 13–20.
[19] Pezhman Sharafi, Bijan Samali, Hamid Ronagh, Maryam Ghodrat, Automated spatial design of multi-story modular buildings using a unified matrix method, Autom. ConStruct. 82 (2017) 31–42.
[20] S. Chaillou, Archigan: artificial intelligence x architecture, in: Architectural Intelligence, Springer, 2020, pp. 117–127.
[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).
[22] P. Merrell, E. Schkufza, V. Koltun, Computer-generated residential building layouts, in: ACM SIGGRAPH Asia 2010 Papers, 2010, pp. 1–12.
[23] N. Nauata, S. Hosseini, K.H. Chang, H. Chu, C.Y. Cheng, Y. Furukawa, House-gan++: generative adversarial layout refinement network towards intelligent computational agent for professional architects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 13632–13641.
[24] Morteza Rahbar, Mohammadjavad Mahdavinejad, Amir HD. Markazi, Mohammadreza Bemanian, Architectural layout design through deep learning and agent-based modeling: a hybrid approach, J. Build. Eng. 47 (2022) 103822.
[25] H. Tanasra, T. Rott Shaham, T. Michaeli, G. Austern, S. Barath, Automation in interior space planning: ultilizing conditional generative adversarial network models to create furniture layouts, Buildings 13–7 (2023) 1793.
[26] L. Wang, J. Liu, Y. Zeng, G. Cheng, H. Hu, J. Hu, X. Huang, Automated building layout generation using deep learning and graph algorithms, Autom. ConStruct. 154 (2023) 105036.
[27] W. Wu, X. Fu, R. Tang, Y. Wang, Y. Qi, L. Liu, Data-driven interior plan generation for residential buildings, ACM Trans. Graph. 38 (6) (2019) 1–12.

[28] Chao-Wang Zhao, Jian Yang, Jiatong Li, Generation of hospital emergency department layouts based on generative adversarial networks, J. Build. Eng. 43 (2021) 102539.
[29] S. Hempel, J. Benner, K.H. Häefele, Generating early design alternatives based on formalized requirements and geospatial data, in: Proceedings of the CIB W, 2015.
[30] R. Sebastian, H. Böhms, P. Bonsma, P. Van Den Helm, 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 (2013) 255–260.
[31] STREAMER-CONSORTIUM, Streamer - european research on energy-efficient healthcare districts, URL: http://www.STREAMER-project.eu/, 2013.
[32] J. Benner, K.H. Häfele, P. Bonsma, M. Bourdeau, S. Soubra, H. Sleiman, S. Robert, Interoperable Tools for Designing Energy-Efficient Buildings in Healthcare Districts, CRC Press/Balkema, 2015.
[33] S. van Nederpelt, K.H. Häfele, S. Hempel, J. Benner, G. Picinbono, J.P. Pols, Parametric modelling techniques for eeb, URL: http://www.STREAMER-project.eu/Downloads/D5.5STREAMER.pdf, 2015.
[34] R. Traversari, M. Den Hoed, R. Di Giulio, F. Bomhof, Towards sustainability through energy efficient buildings design: semantic labels, Entrepreneurship and Sustainability Issues 4 (2017) 243.
[35] M. Weise, T. Liebich, N. Nisbet, State-of-the-art review of advancements and challenges in ontology research, URL: https://www.STREAMER-project.eu/Downloads/D5.1.pdf, 2015.
[36] M. Ester, H.P. Kriegel, J. Sander, X. Xu, et al., A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, kdd, 1996, pp. 226–231.
[37] S. Oprach, T. Bolduan, D. Steuer, M. Vössing, S. Haghsheno, Building the future of the construction industry through artificial intelligence and platform thinking, Digitale Welt 3 (2019) 40–44.
[38] Y. Zhong, A. Geiger, Automated floorplan generation using mathematical optimization, in: ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022, CRC Press, 2023, pp. 404–410.
[39] M. Padberg, G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM Rev. 33 (1991) 60–100.
[40] I. Dikin, Iterative solution of problems of linear and quadratic programming, in: Doklady Akademii Nauk, Russian Academy of Sciences, 1967, pp. 747–748.
[41] K. Bestuzheva, M. Besançon, W.K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, et al., The Scip Optimization Suite 8.0, 2021 arXiv preprint arXiv:2112.08872.
[42] ISO, 16739: 2013 Industry Foundation Classes (Ifc) for Data Sharing in the Construction and Facility Management Industries, vol. 31, International Organization for Standardization, 2013.
[43] K. Madhawa, K. Ishiguro, K. Nakago, M. Abe, Graphnvp: an Invertible Flow Model for Generating Molecular Graphs, 2019 arXiv preprint arXiv:1905.11600.