Optimizing Layout Using Spatial Quality Metrics and User Preferences ☆

Arash Bahrehmand[a,*], Thomas Batard[a], Ricardo Marques[a], Alun Evans[a], Josep Blat[a]

[a] *Department of Information and Communication Technologies (DTIC), Universitat Pompeu Fabra, Barcelona, Spain*

Abstract

Computational design is one of the most common tasks of immersive computer graphics projects, such as games, virtual reality and special effects. Layout planning is a challenging phase of architectural design, which requires optimization across several conflicting criteria. We present an interactive layout solver that assists designers in layout planning by recommending personalized space arrangements based on architectural guidelines and user preferences. Initialized by the designers high-level requirements, an interactive evolutionary algorithm is used to converge on an ideal layout by exploring the space of potential solutions. The major contributions of our proposed approach are addressing subjective aspects of the design to generate personalized layouts; and the development of a genetic algorithm with a multi-parental recombination method that improves the chance of generating higher quality offspring. We demonstrate the ability of our method to generate feasible floor plans which are satisfactory, based on spatial quality metrics and designers taste. The results show that the presented framework can measurably decrease planning complexity by producing layouts which exhibit characteristics of human-made design.

Keywords: layout planning, architectural modeling, genetic algorithm, personalization, interactive systems,

1. Introduction

Computational design is one of the most common tasks of immersive computer graphics projects, such as games, virtual reality and special effects [1, 2]. Layout planning is a significant subcategory of computational design which has recently been the object of research in various layout configuration problems, such as the texture atlases, building arrangement in urban design, furniture arrangement, and interior design [3]. The design of 3D virtual environments in games aims to provide not only a visually engaging experience but also a plausible understanding of the virtual entities through which the user can easily associate real world objects with corresponding 3D digital models [4]. Therefore the digital artist should apply real-world design rules and architectural guidelines to create believable virtual buildings.

Due to the vagueness of the problem knowledge, a solution cannot be clearly formulated at the initial stage. Moreover, there is not a unique best solution for a design problem in architecture, and thus there is likely room to improve the quality of the designs obtained [5]. In addition, given the contextual and subjective nature of solution quality assessment, the steps necessary for determining a solution risk being vaguely defined, and lead to ill-structured solutions [6]. Moreover, creation of virtual layouts can be even more complex and time consuming for the digital artists than for the architects since, first, they normally do not have enough knowledge to address architectural requirements of the building, and secondly, unlike architects who normally concentrate on a single layout at a time, the digital artist may need to create a mass of buildings for a particular district of a virtual city.

This paper proposes an Interactive Layout Recommender System (ILRS) that provides floor layouts to the designer according to his/her preferences and architectural quality metrics. Our ILRS receives high level input constraints from the designer, generates an initial random set of layouts and performs an iterative process to improve the initial population. Our approach takes advantage of evolutionary computing to find solutions which satisfy a balanced account of input constraints. The objective function of our evolutionary algorithm takes into account architectural guidelines as well as users opinion to satisfy both functional and subjective aspects of the design. The proposed system contains an interactive 3D tool that provides the designer with an interface to modify 3D architectural elements such as doors and windows. The target users of our system are digital artists (similarly to [7]) with a basic knowledge of modeling and architecture; however, the system can also benefit architects in early stages of the design process by providing various alternatives based on high level input requirements.

We evaluate ILRS through a detailed comparative analysis of its features with respect to those of several relevant systems and several test experiments. In automatic mode, without user intervention, we test each quality metric separately by weighting it more heavily than the other ones, and show sublinear optimisation improvement and robust termination on scenarios of increasing complexity. With user intervention, we show its ability to produce in a simpler way to actual architectural styles.

Our main novel contributions include:

☆

*Corresponding author
Email address: arash.bahrehmand@upf.edu (Arash Bahrehmand)

- Handling concave and irregular convex shapes as input shapes and output layouts.
- Intelligently generating more accurate initial populations based on dense packing heuristics to decrease the computational costs and to direct the search into a particular region of the solution space which contains high quality solutions.
- Proposing a multi-parental recombination operator to improve the chance of attaining higher quality children that results in the faster convergence towards optimal solutions. It also decreases the probability that the algorithm gets stuck on local optima.
- Applying users opinion in a fitness function in order to satisfy subjective aspects of the design.
- Supporting more spatial quality metrics than the related work, such as circulation and privacy.

Indeed, we provide a simplification that is able to encode some architecturally-inspired style concepts.

The remainder of this paper is structured as follows. First the related work and background concepts are discussed (Sec 2) before describing system overview (Sec 3), problem statement (Sec 4), optimization algorithm (Sec 5), results and evaluation (Sec 6) and concluding the paper by discussion and limitations (Sec 7).

2. Related Work and Background Concepts

Building layouts are mostly created as a result of an iterative trial-and-error process that needs substantial expertise and considerable amount of time [8]. When increasing the number of design factors, space planning turns into a cumbersome task, which makes using specific software more reasonable and practical. Comprehensive literature reviews of automatic layout planners can be found in [9, 10]. The layout solvers have been applied in the following areas: Furniture planning [11, 12], residential space planning [13, 11, 14, 3, 15, 16, 5, 10, 17, 8, 18] and urban design [3, 5]. In the following subsections, we discuss this research categorized according to three criteria: Space unit shape, Optimization model and Recommendation and personalization.

2.1. Space unit shape

Layout planning can be described as deciding on the best arrangement of geometric shapes (i.e. spaces) based on some architectural constraints. We categorise architectural layouts, in terms of the shape format, into three main groups: *Rectangular*, *Rectilinear* and *Arbitrary Polygon*.

A range of solutions have been developed for the synthesis with rectangles as basic units of the input shapes [13, 19, 8, 15, 17]. [9] presented a layout program, EPSAP, to solve a space allocation problem, which only accepts rectangles as input shapes but it is able to generate layouts with rectilinear contour by combining the rectangles and decreasing the compactness level of the arrangement. According to [8], one of the promising lines of research to improve the current state of art would be to add non-rectilinear and curve wall segments. Several studies have applied a top down approach by dividing a base rectangle into some subrectangles, which are interpreted as space units [10, 13, 16]. Some papers discussed matching the arbitrary polygons as a general dense packing problem, while a few of them tried to deal with this issue in architecture design [20, 21]. Authors in [21] state that using non-rectangular modules as pre-designed input shapes is not common in practice, since non-rectangular shapes are generated as the result of combination of input rectangles in the optimization process; the vast majority of algorithms presented use orthogonal polygon shapes as space units in the process of generating layouts; and according to the same source, the use of non convex shapes is absent; [9] claims that most approaches used orthogonal polygon regular shapes as the input modules of the layouts. However, irregular space units and layout boundaries are not unusual in architecture. They can arise from aesthetic considerations of the architect, or even be a requirement if the the building site is not rectangular. Figure 2 [1] shows a real world example of irregular layout from one of the last century's most famous architect.

In order to tackle the limitations of existing systems in terms of dimensional constraints, we augment the range of possible input shapes by supporting irregular polygons resulting in the increase of the diversity of the layouts geometry. This should be useful as well in networks design. Our proposed system accepts deformable predesigned shapes as inputs, and is also able to generate new non-rectilinear shapes through deformation and combination of arbitrary polygons in the evolutionary process.

2.2. Optimization model

The number of problems solved by optimization methods has been increasing. Considering the evolutionary structure of the real world design practice [22], Evolutionary Algorithms (EA) have been used as the backbone of automatic space planning. Recently, Koenig and Schneider [17] applied an EA to generate viable layout solutions without feeding the system with extensive problem specifications. In [18], an EA based system can assist the layout planner by evolving the space topology in an efficient manner. Another hybrid EA technique that benefits from Genetic Algorithms (GAs) to locate and orient residential buildings optimally is used in [14]. [15] used a Simulated Annealing method to generate initial candidate layouts while Merrell [11] applied a similar stochastic optimization method to improve the quality of an individual one. Some approaches take advantage of machine learning techniques, as [8], where an architectural program using a Bayesian Network trained on real world data is proposed. Also, [23] proposes a data driven approach to synthesize the paths between different parts of a mid-scale layout. Our system takes advantage of GA, as a subcategory of EA, to evolve the quality of layouts based on some binary and unary varying techniques.

[1] http://ad009cdnb.archdaily.net/wp-content/uploads/2010/10/1285956198-second-floor-plan.jpg

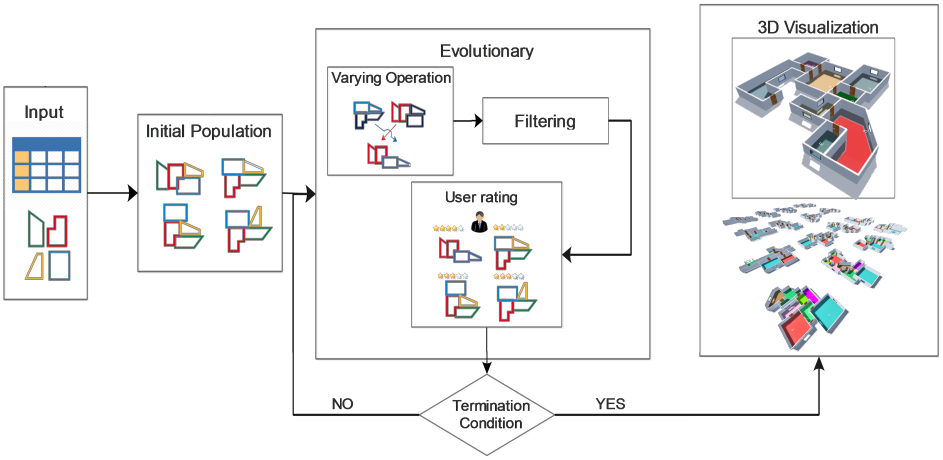


Figure 1: The general scheme of our proposed system (ILRS) that contains four main stages: **Input**, **Initial Population**, **Evolutionary** and **3D Visualization**. The user specifies input constraints at the input stage. In the second stage a semi-random initial population is generated. At the evolutionary stage the system updates the initial generation in an iterative process to reach the optimal solution. User may interrupt the system to rate layouts in some generations with the intention of conducting the search process. Once the termination condition is met, the system enters the final stage (3D visualization) in which the user can perform fine tune modifications to the resulting layouts.

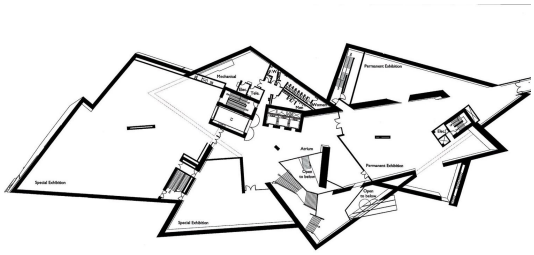


Figure 2: Gallery of Denver Art Museum from Daniel Libeskind

2.3. Recommendation and Personalization

A classical approach that has been applied in several recommender systems is to use item similarity to find the most appropriate item for the user. Applying an interactive evaluation process in evolutionary algorithms already had some successful results as in [24]. However, none of the layout generation systems has applied interactive evaluation to personalize the results. In order to incorporate users preferences in the search process, our system tries to find those layouts which have both good architectural qualities and a certain level of closeness to the users favorite choices. Over several iterations with user feedback, the system accumulates information regarding user preferences. In order to adapt the system to changes in users preferences, the newer layouts in the favorite list are weighted more heavily if the user gets better sense of what s/he wants over time with increasing iterations. In our system, the user can update his/her favorite list at each generation by rating the generated layouts.

3. System Overview

Our system takes as input a set of dimensional and topological constraints, and outputs a set of floor plans through an interactive optimization process. Figure 1 illustrates an overview of the system, which consists of four main stages: specifying the input, generating an initial population, evolutionary optimization and 3D interactive visualization. We present next a brief description of each stage, while Sec 4 and 5 provide a detailed description of the proposed framework along with the mathematical formulation of the problem statement and optimization algorithm.

Input. At this stage the high-level requirements of the designer in the form of dimensional, topological, and opening constraints are organized. An example of input shape can be seen in Figure 3, while Table 1 shows an example of constraints.

Space Unit	$\mathcal{M}$	ϵ	W	D	E
A	I(0.75), J(1.0)	%25	*	J	
B	C(0.5), G(0.25)	%20			
C	B(0.5)	%25			
D		0.50	*		
E	J(0.75)	%25		G	
F	K(0)	%20	*		
G	B(0.25)	%15		E	
H		0.25	*		
I	A(0.75)	%50	*		*
J	A(1.0), E(0.75)	%25		A	
K	F(0)	%50			

Table 1: An example of input preferences, where $\mathcal{M}$ denotes the adjacencies, ϵ is area flexibility determining to which extent the area of a shape can be changed after deformation, **W** is window, **D** is Door and **E** is entrance.

Initial population. The initial population of the evolutionary stage is generated based on a heuristic method. In order to generate a layout in the initial population stage, a copy set of input space units is created and each space unit in the set is randomly deformed based on the area flexibility factor that is defined by the user. Layouts in the initial population satisfy a set of constraints (see Sec 4.2). Each layout is created through semi-random attachment of space units.

Evolutionary. At this stage the system updates the initial population in an iterative process until the termination condi-

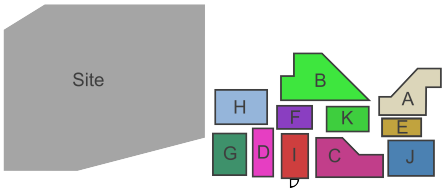


Figure 3: Input shapes of the first experimental analysis. The main entrance is defined in space unit *I*

tion is met. Each iteration in the evolutionary optimization includes two main stages: generating offspring and filtering the best layouts. Offspring are generated through recombination of parent layouts. In order to increase the efficiency of varying methods, a parent selection method (detailed in Sec 5.4.1) is defined. It attempts to select the subset of layouts that more likely generate high quality offspring, which are then used for recombination. In the filtering process layouts with the lower quality are removed from the population. The quality of a layout is measured based on a fitness function (see Sec 5.4.2) that takes into account spatial quality metrics (e.g., circulation, privacy, and compactness), topological relationships between space units, overflow and user ratings (Sec 4.3).

3D interactive visualizer. At this stage, the user is provided with 3D interactive tools to visualize the layouts and modify/refine 3D architectural elements. For example the user is allowed to add, remove openings and change the style of architectural elements based on his/her taste.

4. Problem Statement

Our proposed system produces layout plans by maximizing a layout quality function (which includes user's specifications) over the space of valid layouts. The space of valid layouts is created according to the user's initial preferences defined at the input stage. The notation used for the problem formulation and the proposed optimization algorithm, along with a short explanation, can be found in Table 2.

4.1. Problem Inputs

The input of the system is a set of s closed 2D polygons (the space units) $\mathcal{P}_1, \cdots, \mathcal{P}_s$ as well as a $s \times 6$ *input preferences* table (e.g., Table 1), which indicates the initial preferences of the user in terms of polygon shape and deformation flexibility.

Adjacencies ($\mathcal{M}$ column in Table 1) are encoded in a symmetric $m \times m$ matrix, $\mathcal{M}$, where elements with 0 value indicate that the user does not want the polygons $\mathcal{P}_i$ and $\mathcal{P}_j$ to be connected, 1 means that s/he wants them to be connected, while a value between 0 and 1 determines the user's level of interest to have the corresponding space units as neighbors. Two space units can be adjacent but not necessarily *connected* through a door. The system assigns a random value to those table elements not specified by the user. For instance, the system may randomly add a door between adjacent space units for which the user has not required it. The value ϵ_i for $i = 1, .., m$ determines how much the area of the shape $\mathcal{P}_i$ can be deformed during optimization. Finally, W, D and E specify the user preferences about windows, doors and main entrance, respectively. The user can explicitly determine the existence of doors between space units, windows on space units' walls and the main entrance on space units that are on the wall boundaries.

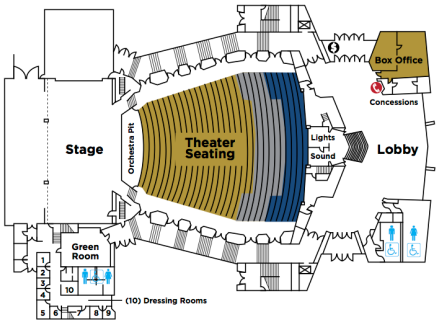


Figure 4: "Theater seating" room should have a particular shape

4.2. Problem Constraints

We require the layouts generated by the system to satisfy four constraints: geometrical, overlap, connectivity, and opening. Formally, we define a layout $\mathcal{L}_a$ as the union of a finite number of closed 2D polygons $\mathcal{P}$, and denote by $\mathcal{S}_\mathcal{L}$ the set of all possible layouts. We also denote by $\mathcal{S}_\mathcal{P}$ the set of arbitrary 2D closed polygons.

Geometrical constraint. The final layout shape results from different factors such as the characteristics of the building site, the arrangement constraints, and (of course) the architect's desired geometrical shape for a particular space unit. For instance, when designing a theater, the architect might envisage a trapezoidal shape for the seating room (Fig. 4)[2]. The shape of the surrounding space units is then based on that of the main room and on that of the site boundary. Sometimes the architect might not have a precise idea about the shape of some of the space units, whose geometry would then be derived in the process from architectural qualities.

Inspired by the concept of deformable shapes in [3] we define *geometrical constraint*, $\phi_\mathcal{G}(\mathcal{L}_a)$, to check if the polygons $\mathcal{P}_a^1, ..., \mathcal{P}_a^m$ of a layout $\mathcal{L}_a$ are within the deformation limits set by the user in the input stage. It is defined as:

$$\phi_\mathcal{G}(\mathcal{L}_a) = \begin{cases} 0 & \text{if } \dfrac{|Area(\mathcal{P}_a^i) - Area(\mathcal{P}^i)|}{Area(\mathcal{P}^i)} < \epsilon_i \qquad \forall i \\ 1 & \text{otherwise} \end{cases}$$

where $Area(\mathcal{P}^i)$ is the area of polygon $\mathcal{P}^i$ as specified by the user, and ϵ_i is the corresponding deformation flexibility.

Connectivity constraint. Connectivity constraints are usually an important factor when evaluating the quality of a given layout (e.g., in [25, 26, 27]). In our framework, the *connectivity*

[2] www.sacramentoconventioncenter.com/Facility/CommunityCenterTheater

Table 2: Symbols applied in problem formulation of the optimization algorithm.

Name	Symbol	Definition
Geometrical constraint	$\phi_{\mathcal{G}}(\mathcal{L}_a)$	Function used to determine if the layout $\mathcal{L}_a$ respects the deformation constraints.
Connectivity constraint	ϕ_{Cv}	It is satisfied if there is at least a path between all possible space pairs.
Opening constraints	$\phi_{Op}(\mathcal{L}_a)$	It is satisfied if the three opening constraints ϕ_D, ϕ_E and ϕ_W are verified
Attachment constraint	$\phi_{\mathcal{A}tt}(\mathcal{L}_a)$	It allows the user to manually attach some space units.
Valid Layouts	$\mathcal{S}_{\mathcal{V}}$	A set of layouts that satisfy hard constraints
Topological Quality	$\phi_{topo}(\mathcal{L}_a)$	To which extent the input topological constraints are satisfied in a generated layout.
Architectural Quality	$\phi_{arch}(\mathcal{L}_a)$	Evaluating the spatial quality of a layout based on some architectural guidelines.
Circulation Quality	$C(\mathcal{L}_a)$	Measuring the quality of a layout based on circulation metric.
Privacy Quality	$\mathcal{P}r(\mathcal{L}_a)$	Measuring the quality of a layout based on Privacy.
Compactness Quality	$Cp(\mathcal{L}_a)$	Measuring the quality of a layout based on compactness.
Weight of Rating	w_u^g	An increasing function of the rating of the user.
Attachment Operator	$\widehat{\cup}(\mathcal{L}_a, \mathcal{L}_b)$	Operator used to attach two given valid layouts $\mathcal{L}_a$ and $\mathcal{L}_b$ by snapping edges.
Candidate Set	$Cndt^g$	A set of candidate sets of compatible sub layouts at generation g.
Sub layout	$l_{i,j}^g$	Sub layout i, j.
Fitness function	$\mathcal{F}^g(\mathcal{L})$	Function which returns the overall quality the layout $\mathcal{L}$ at each generation g.

constraint, ϕ_{Cv}, is satisfied when the layout is a connected set, meaning that there should be a physical path between all possible space unit pairs. ILRS only considers connected layouts, and if a floor plan contained two disconnected layouts, each should be treated as a separate one.

Opening. Three types of opening constraints are defined: door-related constraints (ϕ_D), window-related constraints (ϕ_W) and the main entrance constraint (ϕ_E). All these constraints are represented by binary functions. $\phi_D(\mathcal{L}_a)$ yields 1 if the layout L_a contains all the door connections as required by the user, and 0 otherwise. Similarly, $\phi_W(\mathcal{L}_a) = 1$ if $\mathcal{L}_a$ contains the windows required by the user, and 0 otherwise. Finally, $\phi_E(\mathcal{L}_a)$ returns 1 if the main door location respects the user specification, and 0 otherwise. These constraints can be combined into a ϕ_{Op} constraint, which is 0 if all of them are, and 1 otherwise.

Attachment constraint. The system allows the user to manually attach some space units before the optimization stage, represented as an additional constraint. During the optimization process, the attachment constraint, $\phi_{\mathcal{A}tt}$ is fulfilled if the layout contains the predefined attachment between space units.

We combine the four constraints through a single one $\phi_{\mathcal{H}}(\mathcal{L}_a)$ given by:

$$\phi_{\mathcal{H}}(\mathcal{L}_a) = \begin{cases} 0 & \phi_{\mathcal{G}} = \phi_{Cv} = \phi_{Op} = \phi_{\mathcal{A}tt} = 0 \\ 1 & \text{otherwise} \end{cases}$$

(where we remove the argument $\mathcal{L}_a$ of $\phi_{\mathcal{G}}$, ϕ_{Cv}, ϕ_{Op} and $\phi_{\mathcal{A}tt}$ to simplify notation). Using $\phi_{\mathcal{H}}$ we can now define the set of valid layouts $\mathcal{S}_{\mathcal{V}}$:

$$\mathcal{S}_{\mathcal{V}} := \{\mathcal{L}_a \in \mathcal{S}_{\mathcal{L}} \text{ s.t } \phi_{\mathcal{H}}(\mathcal{L}_a) = 0\}$$

as the set of all layouts $\mathcal{L}_a$ which respect the user defined constraints. Packaging constraints makes ILRS more flexible as other types of constraints could be added.

4.3. Quality metrics

Unlike constraints, which should be fulfilled, quality metrics can take values in $[0, 1]$, where 1 denotes highest quality. Four quality functions are defined: overflow quality, topological quality, spatial quality, and user rating (the latter allowed to take discrete values between 1 and 5).

4.3.1. Overflow quality function

Our layout solver attempts to avoid the overflow of the layout with respect to the site boundaries. To this end, we search for a rigid transformation of the layout (i.e., its shape is preserved) such that it does not exceed the site boundaries. However, finding the best fitting of an arbitrary shape inside another one is not trivial. Indeed, in certain cases, such a transformation might not even exit. We address this problem by applying a heuristic method that is formulated as the search of the rigid transformation that minimizes the area of a layout $\mathcal{L}_a$ beyond the boundaries of the site Ω. Moreover, this heuristic uses the observation that the optimal transformation is more likely found by first mapping the barycenter $c(\mathcal{L}_a)$ of $\mathcal{L}_a$ to the one $c(\Omega)$ of Ω through a translation t, and then finding the rotation that minimizes the exceeding area.

4.3.2. Topological quality function

The adjacency matrix provided as input reflects a major concern of architects when designing layouts, namely, the accessibility between specific space units, which has been discussed in [25, 13]. In [27] the influence of topological attributes of the building layouts on the flow of people is discussed. The topological quality function we define aims at satisfying these accessibility requisites. Given a layout $\mathcal{L}_a = \bigcup_{i=1}^{m} \mathcal{P}_a^i$, its adjacency matrix $\mathcal{M}^a$ is a $m \times m$ matrix that indicates, for each polygon $\mathcal{P}_a^i$, the list of polygons that share at least one edge with it. We define the topological quality $\phi_{topo}(\mathcal{L}_a)$ of $\mathcal{L}_a$ as a measure of the closeness between the adjacency matrix $\mathcal{M}^a$ of

$\mathcal{L}_a$ and the adjacency matrix $\mathcal{M}$ defined by the user's input:

$$\phi_{topo}(\mathcal{L}_a) = \begin{cases} 0 & \text{if } \exists i \neq j \; s.t. \; \mathcal{M}_{ij} = 1 \; and \; \mathcal{M}^a_{ij} = 0 \\ 0 & \text{if } \exists i \neq j \; s.t. \; \mathcal{M}_{ij} = 0 \; and \; \mathcal{M}^a_{ij} = 1 \\ f & \text{otherwise} \end{cases} \quad (1)$$

where

$$f = 1 - \frac{1}{m(m-1)} \sqrt{\sum_{i,j=1}^{m} (\mathcal{M}^a_{ij} - \mathcal{M}_{ij})^2} \quad (2)$$

so that the 0 or 1 requirements must hold, and otherwise the quality is given by f, the distance between the two matrices.

4.3.3. Spatial quality metrics

Within the function ϕ_{arch} we evaluate the spatial quality of a layout based on several architectural guidelines, namely, circulation, privacy and compactness. More precisely, we define the function ϕ_{arch} as

$$\phi_{arch}(\mathcal{L}_a) = w_{\mathcal{C}}\mathcal{C}(\mathcal{L}_a) + w_{\mathcal{P}r}\mathcal{P}r(\mathcal{L}_a) + w_{\mathcal{C}p}\mathcal{C}p(\mathcal{L}_a) \quad (3)$$

s.t.

$$w_{\mathcal{C}} + w_{\mathcal{P}r} + w_{\mathcal{C}p} = 1,$$

where $\mathcal{C}$, $\mathcal{P}r$, and $\mathcal{C}p$ are the circulation, privacy, and compactness quality functions respectively, which are described next.

Circulation. A circulation path defines a semantic relationship between two spatial units in a floorplan. The circulation quality of a layout is measured based on the study presented in [25] which takes into account the path length, path complexity and traffic for each path in the layout. However, in ILRS, we take into account paths connecting all possible points of space units instead of only considering the centers of space units.

Privacy. Privacy determines how and to what extent information about an individual or a group should be communicated to others [28]. Visual privacy is related to a point p in an interior space being (in)visible from a neighbouring space [2]. The point p is visible from another point q if either p and q are in the same convex space unit or there is a direct path between p and q which crosses some windows or doors without any intersection with colliders. To evaluate the visual privacy of a layout, we apply a tessellation based approach to convert the continuous physical space of the floor layouts to a discrete space. The tessellation method creates a grid of cells that have the exact same shape and size. The privacy value is then measured at the center of each cell, separately.

Compactness. The compactness quality defines how efficient a layout is in terms of avoiding the creation of gaps between space units. Formally, the compactness level of a layout $\mathcal{L}_a$ that contains $m > 1$ space units is defined as:

$$\mathcal{C}p(\mathcal{L}_a) = (\text{Number of fit attachments})/(m-1) \quad (4)$$

meaning that less gaps between space units result into more compact layout. However, as noted in [29], sometimes an architect might want an empty space or gap with a particular coordination and orientation in the layout. Our framework supports such intentional gaps by considering them space units, possibly with a high deformation threshold.

4.3.4. User's rating function

Our system allows the user to rate a given layout $\mathcal{L}_a$ through a *Rating* function such that $Rating(\mathcal{L}_a) \in \{1, 2, 3, 4, 5\}$, the score 5 indicating the user is fully satisfied by the layout whereas 1 means not at all.

4.4. Problem formulation

Our goal is to design a system capable of generating the set of optimal valid layouts S_{opt} according to the quality metrics and constraints defined above. Such a problem can be formulated as:

$$S_{opt} = \{\mathcal{L} \in \mathcal{S}_v; Rating(\mathcal{L}) = 5, \\ \phi_{over}(\mathcal{L}) = \phi_{topo}(\mathcal{L}) = \phi_{arch}(\mathcal{L}) = 1\} \quad (5)$$

However, it is not guaranteed that such layouts, which maximize all the desired quality factors, can be found in a reasonable time due to the contradictory nature of the quality metrics and the huge number of possible solutions in the solution space. Our proposal is then to design an algorithm that aims to output a set of layouts $\widetilde{S_{opt}}$ approximating the optimal set S_{opt}, in the sense that the obtained layouts have high user's *Rating* and high quality scores ϕ_{over}, ϕ_{topo} and ϕ_{arch}. To this end, we propose an optimization algorithm based on an evolutionary procedure such that the set of layouts generated at each iteration has not less quality (in a sense that we will detail in Sec. 5) than the ones generated at the previous iteration.

5. Optimization Algorithm

The proposed optimization algorithm has two main phases: the generation of semi-random initial solutions through heuristic techniques; and the evolution of the solutions resulting from the first phase through an evolutionary optimization algorithm. The evolutionary algorithm consists of an iterative process in which a set of offspring layouts is generated at each iteration. These offspring layouts result from combination of two or more parent layouts, in an attempt to produce new layouts with better quality by leveraging the most interesting features of the parents. The best layouts are then selected to be parents in the next iteration from the union set of parents and offspring layouts. This process is repeated until an overall layouts quality criterion is met.

Offspring layouts are constructed through applying varying operators on parent layouts. Figure 5 illustrates an example of the evolutionary algorithm pipeline. For the sake of simplicity, we consider as input polygons, $\mathcal{P}$, squares, and as valid arrangements any horizontal order of squares. The proposed evolutionary algorithm starts from a set $\widetilde{S^0_{opt}}$ of $\mu(> 1)$ valid layouts (see Sec 5.3); μ is the size of population which is 6 in this example. Thus, given $\widetilde{S^g_{opt}}$ as the set of arrangements at generation

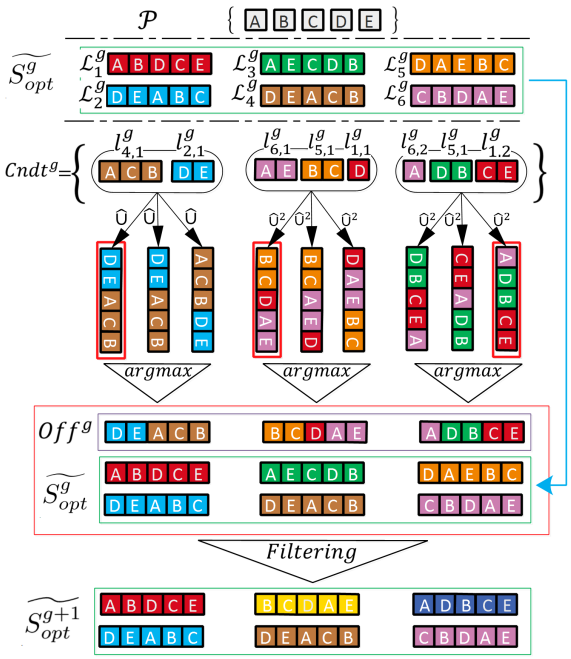


Figure 5: $\mathcal{P}$ is the set of input polygons and $\widetilde{S^g_{opt}}$ is the set of layouts in generation g. Each of the three candidate sets generates three children by applying three times an attachment operator $\widetilde{U}^k, k = 1, 2$. The best offspring of each candidate set is then added to the current list of layouts $\widetilde{S^g_{opt}}$. Finally, six layouts from $\widetilde{S^g_{opt}} \cup Off^g$ are selected for the $g+1$-th generation.

g, $\widetilde{S^{(g+1)}_{opt}}$ is the set of evolved layouts in the generation $g+1$ after filtering out the best layouts at generation g. Filtering is performed based on a fitness function $\mathcal{F}^g$ that is discussed in more detail in 5.1. Thus,

$$\widetilde{\mathcal{S}^{g+1}_{opt}} = \underset{\substack{\mathcal{L}=\{\mathcal{L}_i\}\in(S_{\mathcal{V}})^\mu \\ \mathcal{L}\subset\widetilde{S^g_{opt}}\cup Off^g}}{\arg\max} \sum_{i=1}^{\mu} \mathcal{F}^g(\mathcal{L}_i). \tag{6}$$

where Off^g denotes the offspring generated at generation g. In Figure 5 three offspring at generation g as the consequence of applying varying operator are generated. $S_{\mathcal{V}}$ is the set of valid layouts and $\mathcal{L}_i$ is a layout from the population $\mathcal{L}$ at generation g . $\sum_{i=1}^{\mu} \mathcal{F}^g(\mathcal{L}_i)$ determines the total quality of a set of layouts. It is worth nothing that the total quality of layouts at generation $g+1$ is not less than generation g.

In the following subsections, the different components of the proposed optimization algorithm are presented: construction of fitness function (5.1), user's relative taste function (5.2), attaching operator (5.3) and generating the initial population (5.3).

5.1. Construction of the fitness function

We consider the fitness function $\mathcal{F}^g$ as a linear combination of the quality functions defined in Sect. 4.3 and a user's relative taste function ϕ^g_u (that we define in Sect. 5.2). It is defined as:

$$\begin{aligned}\mathcal{F}^g(\mathcal{L}_a) =& (1 - w^g_u)\,[w_{over}\,\phi_{over}(\mathcal{L}_a) + w_{topo}\,\phi_{topo}(\mathcal{L}_a) \\ &+ w_{arch}\,\phi_{arch}(\mathcal{L}_a)] + w^g_u\,\phi^g_u(\mathcal{L}_a)\end{aligned} \tag{7}$$

where

$$w_{over}+w_{topo}+w_{arch} = 1 \qquad \text{and} \qquad 0 \le w_{over}, w_{topo}, w_{arch}, w^g_u \le 1$$

To address the contextual and subjective nature of solution quality assessment, we allow the user to rate some layouts among $\widetilde{S^g_{opt}}$ at some generation g through the Rating function of 4.3.4. Since, over time, the user's understanding of the best properties of the final goal can change, the system should be able to update the value w^g_u whenever the user rates new items. We set the value of w^g_u to zero in the first generation since the system has not received any feedback from the user. The user may interrupt the iterative process at some iterations to rate. The more the user rates the selected layouts favorably, the more the fitness function of a given layout should take into account the user's input rather than architectural constraints. Therefore, we make the weight w^g_u an increasing function. Indeed, we propose a function

$$w^g_u = \sqrt[n]{\overline{Rate^g}} \tag{8}$$

where $\overline{Rate^g}$ is the average of user's ratings at generation g, for $n \in \mathbb{N}$. As the ratings (i.e., the amount of user feedback the system has received so far) do not decrease throughout the generations, we have that the weight w^g_u is a non-decreasing function of g.

$\mathcal{F}^g$ is applied for all the layouts at each generation to find higher quality solutions and filter low quality solutions. Thus, layouts at each generation are at least as good as the ones selected at the previous generations. Therefore, it can be concluded that as the system runs more generations, the probability of survival of the user's favorite layouts is increased. Ideally, at some generation $\underline{g}$, the weight w^g_u should approach the value 1, meaning that the user is fully satisfied by at least one of the layouts at the generation $\underline{g}$. However, in order to guarantee termination of our evolutionary algorithm, we consider a maximal number of iterations g_{max} for the iterative procedure (6), and we stop it after g^* iterations, where

$$g^* = \min(g_{max}, \underline{g})$$

The set $\widetilde{S_{opt}}$ approximating the optimal set S_{opt} (Eq. (5)) is the set of layouts having the highest fitness at the generation g^*, i.e.:

$$\widetilde{S_{opt}} = \underset{\mathcal{L}\in\widetilde{S^{g^*}_{opt}}}{\arg\max}\ \mathcal{F}^{g^*}(\mathcal{L}) \tag{9}$$

$\widetilde{S_{opt}}$ contains at least one element and at the most μ elements. It is the proposed approximation of the optimal set S_{opt} (5) that we mentioned in 4.4.

5.2. The user's relative taste function

At the generation g, the user's relative taste function ϕ^g_u evaluates the quality of a layout in $\widetilde{S^g_{opt}} \cup Off_g$. ϕ^g_u is based on the distance (Sim, Eq. (10)) between the current layout and the user's rated layouts at previous generations (i.e., S^g_{User}). Let $\mathcal{P}^1, \cdots, \mathcal{P}^m$ be the m input polygons and $\mathcal{L}_a, \mathcal{L}_b$ be two layouts

of the form $\mathcal{L}_a = \bigcup_{j=1}^{m} t_{aj}(\mathcal{P}^j)$, and $\mathcal{L}_b = \bigcup_{j=1}^{m} t_{bj}(\mathcal{P}^j)$ where t_{aj}, t_{bj} are deformations tolerated by the user. Assuming that the site Ω is a polygon, we define the similarity $Sim(\mathcal{L}_a, \mathcal{L}_b)$ between $\mathcal{L}_a$ and $\mathcal{L}_b$ as

$$Sim(\mathcal{L}_a, \mathcal{L}_b) = 1 - \frac{\frac{1}{m}\sqrt{\sum_{j=1}^{m}[c(t_{aj}(\mathcal{P}^j)) - c(t_{bj}(\mathcal{P}^j))]^2}}{\sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}} \quad (10)$$

where $c(\mathcal{P})$ denotes the barycenter of the polygon $\mathcal{P}$, x_{max} (respectively x_{min}) denotes the maximum (respectively the minimum) among the x-coordinates of the vertices of Ω. In the same way, y_{max} (respectively y_{min}) denotes the maximum (respectively the minimum) among the y-coordinates of the vertices of Ω. The denominator in (10) guarantees that Sim is $[0, 1]$-valued.

Using Eq. (10), we define the user's relative taste $\phi_u^g(\mathcal{L}_a)$ of a layout $\mathcal{L}_a \in \widetilde{S_{opt}^g} \cup Off_g$ as

$$\phi_u^g(\mathcal{L}_a) = Rating(\mathcal{L}_{max})/5 \times Sim(\mathcal{L}_a, \mathcal{L}_{max})$$

where

$$\mathcal{L}_{max} = \underset{\mathcal{L} \in S_{User}^g}{\arg\max}\ Sim(\mathcal{L}_a, \mathcal{L})$$

is the most similar layout to $\mathcal{L}_a$ among the set S_{User}^g of user's rated layouts, and $Rating(\mathcal{L}_{max})$ determines the rating the user has assigned to $\mathcal{L}_{max}$.

5.3. Construction of a layout

The aim of this section is to introduce a recursive attachment operator that generates a semi-random valid layout from a given set of layouts. This operator is applied in the evolutionary process to construct the initial population $\widetilde{S_{opt}^0}$ from the input polygons $\mathcal{P}^1, \cdots, \mathcal{P}^m$ (considering a polygon as a layout with one single space unit), as well as the offspring Off^g at each generation g from the set $\widetilde{S_{opt}^g}$.

Attachment operator. Two given valid layouts are attached together by snapping two edges on their contours so that these layouts have at least one shared edge, either partially or fully shared, after applying the attachment. An *attachment operator* is defined as:

$$\begin{array}{rccc} \widehat{\cup}: & S_{\mathcal{L}} \times S_{\mathcal{L}} & \longrightarrow & S_{\mathcal{L}} \\ & (\mathcal{L}_a, \mathcal{L}_b) & \longmapsto & \mathcal{L}_a \bigcup t_{ab}(\mathcal{L}_b) \end{array}$$

where $t_{ab} \in T$ is a transformation that maps (snaps) a selected point e_{n_b,k_b}^b on the edge $e_{n_b}^b$ of the contour of $\mathcal{L}_b$ to a selected point e_{n_a,k_a}^a on the edge $e_{n_a}^a$ of the contour of $\mathcal{L}_a$. From the attachment operator $\widehat{\cup}$ between two valid layouts, we derive a recursive attachment operator $\widehat{\cup}^{k-1}$ between k valid layouts, as:

$$\begin{array}{rccc} \widehat{\cup}^{k-1}: & (S_{\mathcal{L}})^k & \longrightarrow & S_{\mathcal{L}} \\ & (\mathcal{L}_1, \cdots, \mathcal{L}_k) & \longmapsto & \widehat{\cup}(\mathcal{L}_1, \cdots, \widehat{\cup}(\mathcal{L}_{k-1}, \mathcal{L}_k)\cdots) \end{array} \quad (11)$$

We say that an attachment between the k valid layouts $(\mathcal{L}_1, \cdots, \mathcal{L}_k)$ is *valid* if the output $\widehat{\cup}^{k-1}(\mathcal{L}_1, \cdots, \mathcal{L}_k)$ is a valid layout. Figure 7 illustrates examples of valid and invalid attachments.

Deformation. Layouts might be deformed before or after attachment. Each attachment operand is subjected to a random deformation before applying the attachment operator. The deformation is valid if it does not change the area more than a threshold (see Sec 4.2). The random deformation is performed by random translation of edges or vertices. Attachment may cause intersection between two layouts when at least one of them is concave. In our system, if applying the attachment operator causes an intersection, the intersected region is subtracted from one of the attachment operands.

Random point selection. In order to generate a valid layout, the attachment operator (11) is iterated several times until a valid attachment occurs. To improve the efficiency of the attachment process, a heuristic is applied to reduce the number of iterations by recording the history of previous *invalid* attachments.

Fit attachment. As indicated in Sect. 4.3.3 the compactness level of a layout relies on the number of *fit attachments*. We define a fit attachment between a layout $\mathcal{L}_a$ whose contour is convex and a layout $\mathcal{L}_b$ whose contour is concave as a valid attachment if the attachment points are vertices $e_{n_a,0}^a$ and $e_{n_b,0}^b$ satisfy

$$\theta(e_{n_a,0}^a) = 2\pi - \theta(e_{n_b,0}^b)$$

where θ denotes the internal angle between the two edges joining at a vertex. For instance, both attachments in Fig. 7 are fit attachments.

5.4. Construction of the set $\widetilde{S}_{opt}^{g+1}$ in formula (6)

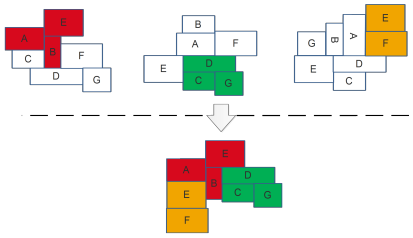


Figure 6: An offspring is generated as the consequence of combining three compatible sub layouts that are differentiated with red, green and orange colors.

Given a set of individuals, EA algorithms attempt to increase the fitness of the population through two main operations: variation and selection. We apply the so called two-tier $(\mu+\lambda)-ES$ as a general form of evolutionary algorithm where μ is the number of individuals that are kept between generations (the sets $\widetilde{S_{opt}^g}$, $g \geq 0$ in Eq. (6)) and λ is the number of individuals that are generated through reproduction in each generation (the offspring set Off^g in Eq. (6)). In the last step, the system sorts the $\lambda+\mu$ layouts in the set $\widetilde{S_{opt}^g} \cup Off^g$ based on their quality scores $\mathcal{F}^g$, defined by Eq. (7), from which the set $\widetilde{S_{opt}^{g+1}}$ is generated.

5.4.1. Varying

In our proposed system, both shape and arrangement of space units are varied. Each offspring is presented as a new solution in the solution space therefore a good representation of the varying operations guarantees the connectivity of the solution space. The recombination operator randomly chooses a part of each parent for the mating process. The resulting offspring is acceptable if it gives higher yields by inheriting desirable features of the parents. Therefore, parent characteristics have a substantial influence on the quality of the offspring which makes the process of parent selection so critical. Usually, in EA, high quality individuals have a higher chance to become parent than low quality individuals. However the chances for low quality individuals should not be too small to avoid getting stuck in a local optimum [30]. We define mutation as a particular type of recombination in which parents are selected from the same layout. The proposed recombination method includes two main stages: selecting a set of parents, and creating an offspring from combining these parents. We then introduce the concept of compatible parents in order to increase the chance of generating a high quality offspring.

Compatibility of a set of layouts. Let $\mathcal{L}^g = \{\mathcal{L}_1^g, \cdots, \mathcal{L}_\mu^g\}$ be a set of μ layouts at the g-th generation. Recall that a layout $\mathcal{L}_i^g$, $i = 1, ...\mu$, in $\mathcal{L}^g$ is of the form

$$\mathcal{L}_i^g = \bigcup_{j=1}^{m} t_{ij}(\mathcal{P}^j)$$

for some deformations t_{ij} applied to the input polygons $\mathcal{P} = \{\mathcal{P}^1, \cdots, \mathcal{P}^m\}$. More precisely, given a random edge $e_{n_b}^b$ in $Cntr(\mathcal{L}_b)$, that belongs to a unique space unit $t_{bj}(\mathcal{P}^j)$, we consider the ordered set of polygons $P^{j_1}, \cdots, P^{j_{k_j}}$ that have an edge in $Cntr(\mathcal{L}_a)$ and whose adjacency with P^j is strictly greater than 0, i.e. such that $\mathcal{M}_{jj_1} > \cdots > \mathcal{M}_{jj_{k_j}} > 0$ where $\mathcal{M}$ is the input adjacency matrix. Then, given a random edge $e_{n_a}^a$ in $Cntr(\mathcal{L}_a)$, that belongs to the space unit $t_{aj_1}(\mathcal{P}^{j_1})$, the system snaps $e_{n_b}^b$ to $e_{n_a}^a$. If this attachment is not valid, the system snaps $e_{n_b}^b$ to a random edge in $Cntr(\mathcal{L}_a)$ that belongs to $t_{aj_2}(\mathcal{P}^{j_2})$, and so on until j_{k_j}. Finally, if the attachment for j_{k_j} is not valid, the system snaps $e_{n_b}^b$ to a random edge in the contour of $\mathcal{L}_a$. Figure 7 shows an example of desired neighbors (determined by the user) for a space unit and a preferable attachment between this space unit, considered as a layout, and another layout.

Parent Selection. We present an intelligent variation method that takes into account the quality and compatibility level of layouts in the process of parent selection. Moreover, in order to increase the chances of generating more creative layouts, our system supports n-ary parent set where $1 < n < m$ (m being the number of input space units). The parent set is the set of parent layouts that participate in the process of recombination. Each parent layout contributes with some of its space units in the recombination process for producing the final set of offspring. The fitness of an offspring does not only depend on the compatibility of the sub layouts that generate it, but also on the fitness $\mathcal{F}^l$ of each sub layout as well. Given a sub layout $l_{i,k}^g$, we define its fitness $\mathcal{F}^l(l_{i,k}^g)$ as

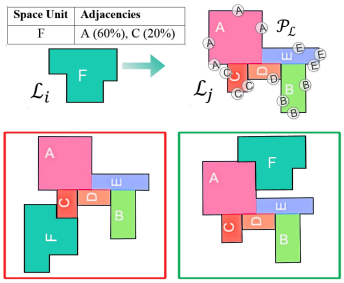


Figure 7: The desired neighbors of F are A and C while the user tends to prefer A to B as the neighbor of F. In the second row, the left figure is a not an appropriate attachment since F is attached to C while there is the possibility of a more appropriate attachment by attaching F to A. The right figure is a preferable attachment.

$$\mathcal{F}^l(l_{i,k}^g) = w_{over}\,\phi_{over}(l_{i,k}^g) + w_{topo}\,\phi_{topo}(l_{i,k}^g) + w_{arch}\,\phi_{arch}(l_{i,k}^g) \quad (12)$$

Where i is the number of input polygons in the sub layout and k is the identifier of a specific sub layout from the list of all possible sub layouts with i polygons. In order to increase the probability of generating the λ children of highest quality, we construct a set $Cndt^g$ of λ *candidates* sets of compatible sub layouts $(Cndt_1^g, \cdots, Cndt_\lambda^g)$ having the highest fitness. The fitness of a set of sub layouts is defined as the sum of the fitness of each sub layout, given by formula (12). The space $Cndt^g$ is given by

$$Cndt^g = \underset{\substack{S=(S_1,\cdots,S_\lambda)\\ S_i=(l_{i_1,k_{i_1}}^g,\cdots,l_{i_{n_i},k_{i_{n_i}}}^g)\\ Cmp(S_i)=1}}{\arg\max} \sum_{i=1}^{\lambda}\sum_{n=1}^{n_i} \mathcal{F}^l(l_{i_n,k_{i_n}}^g) \quad (13)$$

for $i_n \in \{1, \cdots, m\}$, and $1 \le k_{i_1} < \cdots < k_{i_{n_i}} \le 2^m - 1$.

Updating generation. For any candidate set $Cndt_i^g \in Cndt^g, i = 1, \cdots, \lambda$, we generate p valid layouts by attaching the sub layouts in $Cndt_i^g$ through q_i realizations, $q_i \ge p$, of the operator $\widehat{\cup}^{|Cndt_i^g|-1}$ defined in (11), and we select among these p layouts the one having the highest fitness (7), that we call $\mathcal{L}_i^{*g}$. We then consider the offspring set Off^g at the generation g as the set containing these better fit layouts, i.e. $Off^g = (\mathcal{L}_1^{*g}, \cdots, \mathcal{L}_\lambda^{*g})$ where

$$\mathcal{L}_i^{*g} = \underset{q_i \text{ realizations of } \widehat{\cup}^{|Cndt_i^g|-1}}{\arg\max} \mathcal{F}^g\left(\widehat{\cup}^{|Cndt_i^g|-1}(Cndt_i^g)\right) \quad (14)$$

5.4.2. Filtering (Selection)

After generating the offspring, the filtering method is applied to select the best μ solutions. The set $\widetilde{S_{opt}^{g+1}} = (\mathcal{L}_1^{g+1}, \cdots, \mathcal{L}_\mu^{g+1})$ of parent layouts at the generation $g+1$ is determined by the set of the μ layouts among the parents and the offspring at the

generation g having the highest fitness, i.e.

$$\widetilde{S_{opt}^{g+1}} = \underset{\substack{\mathcal{L}=(\mathcal{L}_1,\cdots,\mathcal{L}_\mu) \\ \mathcal{L}\subset \widetilde{S_{opt}^{g}}\cup Off^g}}{\arg\max} \sum_{k=1}^{\mu} \mathcal{F}^g(\mathcal{L}_k) \quad (15)$$

In Figure 5 we illustrate an example of the varying process and selection, including steps given by formulae (13), (14) and (15).

6. Evaluation

Automatic layout solvers are usually evaluated based on the quality, the variety and the validity of the set of layouts generated as outcome. Additionally the efficiency of the solver can be assessed through the number of iterations required to achieve the set of final solutions [31]. The main contribution of our solver is its ability to fully use user's input, which is key in architecture. However, we evaluate first the system in automatic mode. Indeed, in 6.1 we show how each quality metric works by weighting it more heavily, and complement these experiments with others showing sublinear optimisation improvement and robust termination under a variety of conditions. Then, we test real application cases with user intervention in 6.2, especially showing the ability of producing layouts similar to architect's ones from constraints and user's ratings. Finally, we compare the features of ILRS with those of several relevant systems in 6.3.

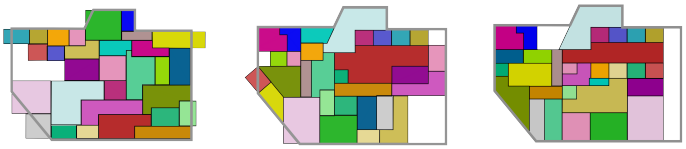
6.1. Results in Automatic Mode

This section presents two sets of experiments where ILRS is used as a fully automatic tool, without user's input. They help to grasp how the algorithm works in practice. In these experiments we set the weight of the user rating to $w_u = 0$ in the fitness function.

6.1.1. The Impact of the Quality Metrics

The evaluation in this subsubsection is inspired by [11], where an understanding of how different quality metrics impact on the results of automatically computed furniture layouts is obtained by showing results obtained by removing one of the quality metrics used in the optimization process. In our case, we only use one quality metric at the time and show results at different stages. This illustrates how each quality metric drives the optimization. Thus, four test scenarios are defined. In each one, one quality metric dominates the other ones by significantly increasing its weight in the fitness function (Eq. (7)). The same set of input shapes (i.e., $|\mathcal{P}| = 30$) is used, the number of iterations g^* is set to 30, and 16 layouts are produced in each generation (i.e., $\mu = 16$). For each scenario we show the layouts with the highest fitness function values in three different generations of the evolution process: the first ($g^* = 1$), an intermediate ($g^* = 15$) and the final one ($g^* = 30$).

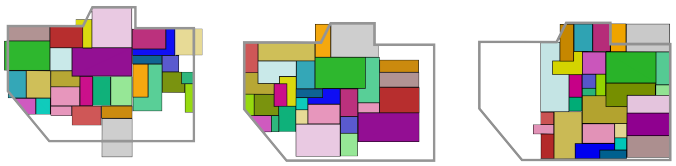
Overflow: Fig. 8 shows the results for the case where the fitness function is dominated by the overflow quality metric. Recall that this metric is used to make the layouts fit the site boundaries. This is precisely the case of the last generation layout shown (Fig. 8 (c)). In early evolutionary phases (Fig. 8 (b)), the layout can exceed the site boundaries. Then, as the evolutionary process goes on, the layouts progressively present less and less overflow.



(a) $g = 1$ (b) $g = 15$ $g = 30$

Figure 8: Layouts from three different generations obtained using a fitness function dominated by the *overflow* quality metrics.

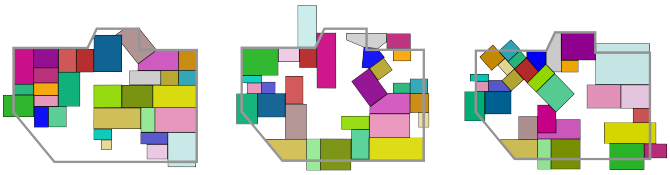
Compactness: The results in Fig. 9, where compactness dominates, show that (c) is more efficient than (b) and (a) in terms of space occupied within the boundaries, as expected. Comparing with with Fig. 8, where overflow dominates, the layouts generated are more compact, with less empty spaces within the boundaries; on the contrary, the final layouts might have overflow.



(a) $g = 1$ (b) $g = 15$ $g = 30$

Figure 9: Layouts from three different generations obtained using a fitness function dominated by the *compactness* quality metrics.

Privacy: This scenario prioritizes privacy over other quality factors. The layouts in Fig. 10 show the evolution towards minimizing the connections between space units and reducing the number of neighbouring space units, and empty spaces emerge within the boundaries of the site. The layouts present less visibility and more privacy than those resulting from previous scenarios.



(a) $g = 1$ (b) $g = 15$ $g = 30$

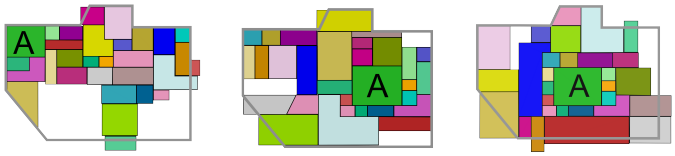
Figure 10: Layouts from three different generations obtained using a fitness function dominated by the *privacy* quality metrics.

Circulation: The impact of the circulation metrics is more difficult to make visible, as it involves a complex setting of space units, which should be easily connected. Thus, we added a simpler topological constraint in the test, namely, that the space unit A should have the maximum number of neighbors among other space units, with the quality metrics W_{topo} having assigned a significant weight. As illustrated in Fig. 11 (c),

Scenario	$\|\mathcal{P}\|$	μ	Topo	$t(\mathcal{L}_i)$	$t(g_n)$	g^*
1	4	16	~30%	0.43	6.71	14
		32	~50%	0.67	21.01	12
2	8	16	~20%	2.18	33.31	25
		32	~40%	3.34	68.11	21
3	16	32	~20%	4.89	89.74	39
		64	~30%	6.53	423.51	28
4	32	32	~15%	29.61	766.96	42
		128	~20%	48.22	7622.18	33
5	64	128	~10%	98.05	19377.48	48
		256	~20%	133.10	26351.14	36

Table 3: Scenarios and results of the performance tests; time in seconds

A has been surrounded by its neighbors so that all of them are accessible by A.



(a) $g = 1$ (b) $g = 15$ $g = 30$

Figure 11: Layouts from three different generations dominated by *circulation* and a topological constraint on A.

6.1.2. Automatic Performance

With ILRS in automatic mode, this set of experiments contemplates five scenarios, each with different geometrical and topological constraints, and two different population sizes, making a total of 10 experiments. The termination condition is met when the overall quality of the best layout does not change more than a threshold value over the last four generations (similarly to [31]). Thus, we have results about robustness and termination with respect to a variety of conditions, optimization improvements, and performance of the system with increasing sizes, which complement those of the previous subsubsection.

Table 3 reflects the parameters used and the results achieved for each experiment: $|\mathcal{P}|$ is the number of space units, μ is the population size, and *Topo* represents the density of adjacency matrix and g^* is the number of generations. The average time, $t(\mathcal{L}_i)$, to generate a single valid layout in the semi-random generation of valid layouts. The average time spent on each generation, $t(g_n)$, creating layouts through recombination, and selecting the fittest layouts. Figures 12, 13, 14, 15, and 16 show examples of layouts of the last generation of each experiment and the graphs of the anytime behavior [32] of our EA. The black contour represents the geometry of the site boundary.

According to the Table 3 and the figures, our EA system improves the overall quality of initial populations in a sublinear way and reaches optimised solutions. The average time of a generation increases with respect to the number of layouts. Indeed, the larger the layouts, the heavier the required computation of the techniques, including parent selection and recombination. The screenshots of final layouts of the last generation of each experiment, and especially, in the case of 64 space units, demonstrate that the system can address in a robust and scalable way for a variety of cases. These set of experiments result into layouts with some jagged edges and some empty spaces. This is caused by the fitness functions chosen, and is not a limitation of our approach. In the previous set of experiments (6.1.1), where a quality factor dominates, and in actual application cases, shown next, where the user intervention can drive the simulation, the results give more compact and regular layouts, if this is desired by the user or intended through the fitness function.

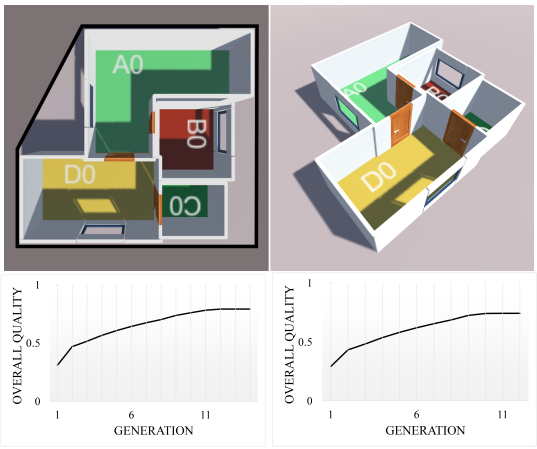


Figure 12: First scenario of performance test

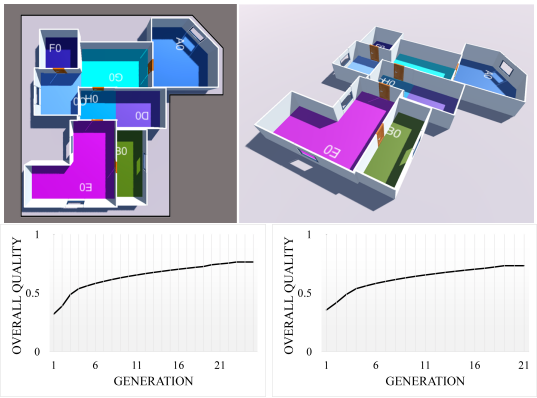


Figure 13: Second scenario of performance test

6.2. Experiments with user intervention

The following two experiments include user's contribution. First, a simulated situation with user intervention is analyzed in detail. Then, the ability to reproduce actual human designs, from constraints and fitness function related to the actual design intended and through user's ratings, is shown.

6.2.1. An illustrative use case in detail: Competence

The experiment presented in this section was performed using the input shapes shown in Figure 3 and the user preferences shown on Table 1. The intended goal was to produce compact layouts with a good circulation quality. Therefore, the weights

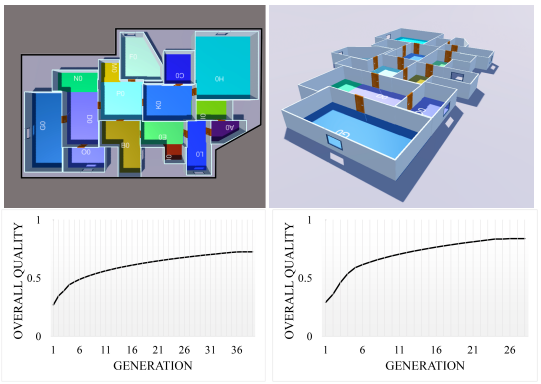


Figure 14: Third scenario of performance test

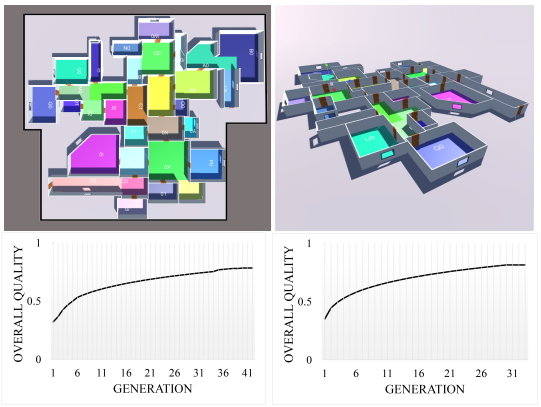


Figure 15: Fourth scenario of performance test

of compactness w_{Cp} and circulation w_C were set to be larger than those corresponding to other factors (some other possible settings are discussed in section 7). The choice of the parameter values used in this experiment is given in table 4. We chose μ based on the performance results of the previous section (see table 3), larger than $2|\mathcal{P}|$ in order to allow reaching optimised solutions in less than $|\mathcal{P}|^2$ generations. However, we should remark that the variation of the number of generations required to reach the optimised solution can be large due to the randomness of the generative process, reinforced if the system presents layouts that do not match the user opinion in the first generations; g^* is the generation number in which the optimised solution is found and $itrp$ is the number of times the user interrupted the system for rating. The termination condition was met when the user assigned five stars rating to at least one layout. The final layout was found in generation 86.

Figure 17 shows the graphs representing the values of the different quality metrics (Y axis) versus the generation number (X axis). We see that:

1. The overall quality increases at each generation and gets closer to the highest quality layout by approaching g^*.

2. The graphs, except for W_u, have sublinear growth, and roughly logarithmic shape.

Figure 18 (a) is a screenshot of nine solutions at the first generation. Figure 19 shows five selected layouts that were rated

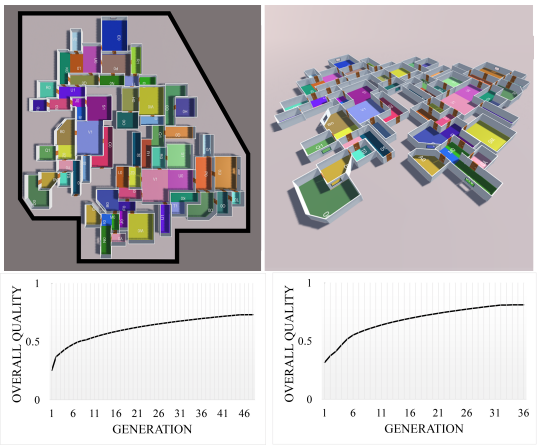


Figure 16: Fifth scenario of performance test

by user in different generations. Figure 18 (b) shows some layouts of the last generation, where layout 6 (highlighted by a red contour) was seen as the layout with the highest score layout. This experiment illustrates that the system can generate various layout solutions where the average quality of individuals in each generation improves until the termination stage.

6.2.2. *Validation Analysis Against Real World Examples*

To validate our approach against real world situations, we ran an experiment with the goal of obtaining layouts as close as possible to an actual floor plan. The reference layout is shown in Figure 20 and was selected from [33]. This led us to define the input constraints based on this target specification, and a suitable fitness function. However, we did not impose to fulfill all the topological constraints of the target, with the intention of increasing the variety of alternative solutions. On the other hand, given that the user knows exactly the layout which is targeted at and looks for similarity, w_u is larger than other weights in the optimization process. Actually, the user stopped the search process for rating 6 times.

Figure 21 shows nine layouts of the last generations. The layout most similar to the reference one is highlighted with a red rectangle, illustrating that ILRS can obtain layouts similar to floor plans made by a human designer from intended constraints and user's ratings. The layouts highlighted with a blue rectangle possess similar architectural style concepts of the reference layout with a slightly different arrangement: this would allow the user to explore variations of an optimised solution.

6.3. *Comparative Analysis*

In this subsection, we compare the features of ILRS with three alternative representative studies which are the most relevant to our approach, namely, Evolutionary Program for Space Allocation Problem (EPSAP [31]), Computer-Generated Residential Building Layout (CGRBL, [8]) and Recommendation of floor plans (REP, [13]).

Table 5 compares the features of the four systems. Given that these systems provide layouts through automatic architectural design, it is especially important that they support rich

μ =32	λ =16	$itrp$ =9	g^* =86	$\|\mathcal{P}\|$ =11	w_{arch} =0.5	w_{Cp} =0.4	w_C =0.4	$w_{\mathcal{P}r}$ =0.2	w_{over} =0.25	w_{topo} =0.25

Table 4: Constraints values of quantitative analysis

	ILRS	**EPSAP**[31]	**CGRBL**[8]	**REP**[13]
Quality Constraints	Sp Ov, Cr, Op Sp, Pr, Cp, Cn, LD, and Of	Sp Ov, Op Or, Op Ov, Cp, Cn, LD, and Of	Sp Ov , Cn, LD and Of	Sp Ov, Cn, LD, Op Or, and Of
Attachment Constraint	Supported	Not Supported	Not Supported	Supported
Shape Variety	Arbitrary Polygon	Rectilinear Polygons	Rectilinear Polygons	Rectilinear Polygons
Interaction before Optimization	Specify Input requirements	Specify Input requirements	Specify Input requirements	Specify Input requirements
Interaction during Optimization	Rating Layouts	No Interaction	No Interaction	No Interaction
Interaction after Optimization	Fine tune modifications	No Interaction	No Interaction	No Interaction

Table 5: Features Comparison. SO(Space Overlap), Op Or(Opening Orientation), Op Ov(Opening Overlap), Op Sp(Opening Space Unit), Cp(Compactness), Cn(Connectivity), LD(Layout Dimension), Of(Overflow), Cr(Circulation) and Pr(Privacy).

metrics of architectural quality of the produced layouts. In this respect, ILRS and EPSAP are the systems which support more quality constraints, with ILRS supporting circulation and privacy beyond EPSAP. Only ILRS and REP support attachment constraints meaning that they include essential users requirements from the initial stage, and allow to filter out irrelevant arrangements from the space of solutions, thus saving computation time. All the systems support initial input requirements provided by users. However, ILRS is the only one where the designer can interact during the layout generation process to guide the search process towards layouts that match better the users tastes. Additionally, ILRS supports fine tuning of the final results, which makes it more applicable. ILRS is the only system allowing for arbitrary polygons, which means an additional style flexibility.

7. Discussion, Limitations and Future Work

In Sec 6.1.1 it has been shown that, when executed in automatic mode, ILRS is able to produce layouts with different features, depending on the weights attributed to each quality metrics. In Sec. 6.1.2, measures of the path to optimality and termination with respect to a variety of conditions were provided. The comparative analysis of Sec. 6.3 shows that ILRS is more flexible and supports a wider variety of architectural qualities than related relevant systems.

However, one of the most distinctive contributions of ILRS, is its ability to produce layouts which are driven by the user's tastes. Moreover, as illustrated in Sec. 6.2.2, RTIS is also capable of producing (a variety of) layouts with similar characteristics to architect made floor plans, through a combination of input constraints and user's ratings. This ability to provide to the user a set of layouts which comply with the input constraints and that, at the same time, follow the user's subjective preferences (which can even be adjusted during the process), represents a major advantage of RTIS. Indeed, when compared to the effort demanded to a digital artist for the manual production of layouts, the user rating process, the definition of input constraints and of the fitness function required by RTIS can be considered relatively minor tasks.

It is interesting to discuss the meaning of the parameters of the fitness function. Take as an illustrative example the floor plan of a hospital: the physical relationships or connections between space units play an essential role in transferring patients and medical instruments. Therefore, the designer may give a higher weight to circulation quality. Hospitals are usually built at sites which are larger than the built area. This gives the designer more freedom to find an optimised arrangement provided it does not exceed the site boundary. Therefore, the compactness should be attributed a lower weight, while overflow will be a constraint. When privacy is important, this would force the designer to set up the openings, especially windows, very precisely, and to increase the weight of privacy. In contrast, when designing the layout of a prison, the room visibility dimension, which is somehow the opposite of visual privacy, should have a high value. In general, input parameters such as connections, or area flexibility and weights of quality metrics in the fitness function define the characteristics of the layout solutions that our system generates.

Let us give some examples of input parameters as well. If the user is searching for the best arrangement of a set of known space units (in terms of geometrical attributes), as is usually the case in procedural generation for video game design, s/he may set the area flexibility of all input space units to zero and leave the topological column empty. On the other hand, if the designer wanted to explore possible shapes of a finite number of space units which satisfy a particular topological matrix, then s/he would give high values to the area flexibility and set the adjacency values to 1 according to the topological matrix. Then the solution space is reduced to layouts which satisfy the topological matrix and the area flexibility thresholds. The designer's ratings would drive further the exploration.

Some limitations of our proposal, which lead to issues that are of interest for future research are:

- The set of input constraints required and the appropriate fitness function. In systems such as ours, the way of defining the input constraints, and establishing the weights of a fitness function is very unfamiliar when considering what architects are used to work with. Designers (and architects

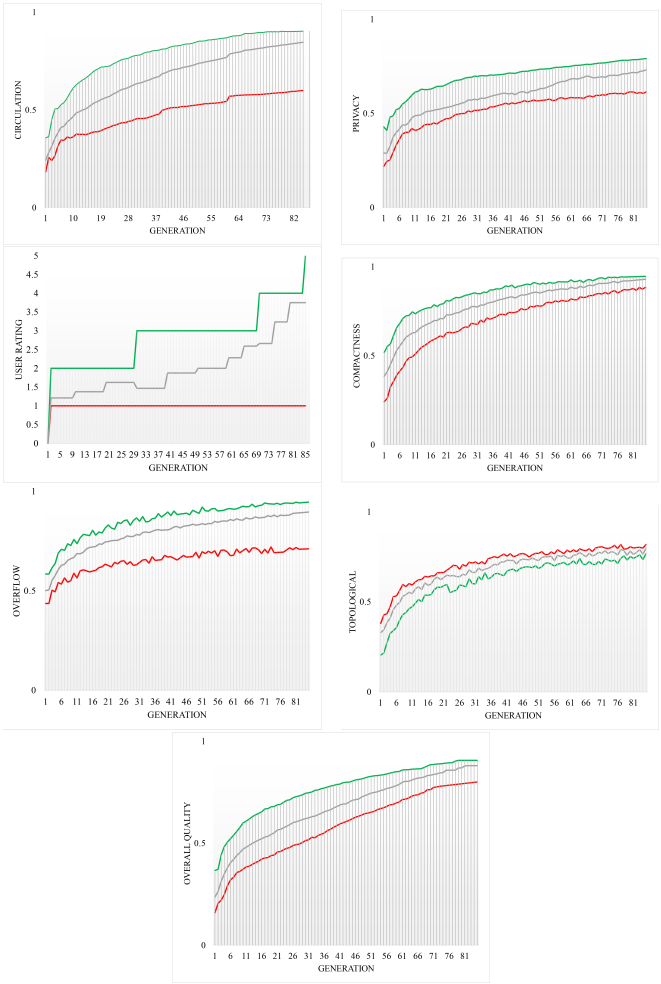


Figure 17: Quality statistics of the competence experiment. The green color represents the highest quality, the red represents the lowest quality and the gray is the avarage quality at each generation.

too) normally prefer to deal with more abstract and less complex tools. In future work, we plan to provide the architect/designer with some semantic presets which determine the specific values of the constraints, and the fitness function to achieve some predefined layout types such as residential, hospital or school. The constraints and parameters might be learned from suitable datasets.

- Determining the contribution of the user's ratings at each generation. Learning the user's ratings seems quite difficult, as it would require suitable data sets; and it seems difficult and impractical to learn them from ILRS itself, because of the smallness of the data set, and the possible change of perspectives of the designer during the process. On the other hand, in Eq. (8), n defines the slope of a function that sets how fast the user contribution is raised over generations. While a larger value of n results in faster convergence, it leads as well to the creation of layouts similar to user selected items, but which are not necessarily high quality in terms of architectural qualities. On the other hand, a lower value of n causes slower convergence and generating layouts with higher architectural quality but not necessary what the user would expect. Improving this is a research challenge.
- Finally, the layouts in our system are currently generated at a single level; 2D layouts can be extruded to provide a 3D view of the floor plan to the user, which is supported by our system. In Figures 12 to 16 the 3D layouts are generated automatically from the 2D layout. A research avenue is to support floorplans with multiple levels (above or under ground) so that designers could handle more complex scenarios.

References

[1] Lobos, D., Donath, D.. The problem of space layout in architecture: A survey and reflections. 2010.

[2] Indraprastha, A., Shinozaki, M.. Computational models for measuring spatial quality of interior design in virtual environment. Building and Environment 2012; 49:67–85. doi:10.1016/j.buildenv.2011.09.017.

[3] Peng, C.H., Yang, Y.L., Wonka, P.. Computing layouts with deformable templates. ACM Transactions on Graphics (TOG) 2014; 33(4):99.

[4] Vasin, Y., Osipov, M.P., Muntyan, S.V., Kustov, E.A.. Procedural modeling and interactive 3D visualization of objects of the internal structure of buildings and facilities. Pattern Recognition and Image Analysis 2015; 25(2):278–280. doi:10.1134/S105466181502025X.

[5] Homayouni, H.. A Survey of Computational Approaches to Space Layout Planning (1965-2000). Department of Architecture and Urban Planning University of Washington 2000;.

[6] Simon, H.A.. The structure of ill-structured problems. In: Models of discovery. Springer; 1977, p. 304–325.

[7] Leblanc, L., Houle, J., Poulin, P.. Component-based modeling of complete buildings. In: Proceedings of Graphics Interface 2011. Canadian Human-Computer Communications Society; 2011, p. 87–94.

[8] Merrell, P., Schkufza, E., Koltun, V.. Computer-generated residential building layouts. In: ACM SIGGRAPH Asia 2010 papers. SIGGRAPH ASIA '10; New York, NY, USA: ACM. ISBN 978-1-4503-0439-9; 2010, p. 181:1—-181:12. doi:10.1145/1866158.1866203.

[9] Rodrigues, E., Gaspar, A.R., Gomes, Á.. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. Computer-Aided Design 2013; 45(5):887–897.

[10] Koenig, R., Knecht, K.. Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 2014; 28(03):285–299.

[11] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., Koltun, V.. Interactive furniture layout using interior design guidelines. In: ACM SIGGRAPH 2011 papers. SIGGRAPH '11; New York, NY, USA: ACM. ISBN 978-1-4503-0943-1; 2011, p. 87:1—-87:10. doi:10.1145/1964921.1964982.

[12] Yu, L.F., Yeung, S.K., Tang, C.K., Terzopoulos, D., Chan, T.F., Osher, S.J.. Make it home: automatic optimization of furniture arrangement 2011;.

[13] Bahrehmand, A., Evans, A., Blat, J.. Recommendation of Floor Plan Layouts Based on Binary Trees. In: egice. Cardiff; 2014, p. 1–10.

[14] Ansary, A.M.E., Shalaby, M.F.. Evolutionary optimization technique for site layout planning. Sustainable Cities and Society 2014; 11:48–55. doi:http://dx.doi.org/10.1016/j.scs.2013.11.008.

[15] Bao, F., Yan, D.M., Mitra, N.J., Wonka, P.. Generating and exploring good building layouts. ACM Transactions on Graphics (TOG) 2013; 32(4):122.

[16] Knecht, K., Koenig, R.. Generating floor plan layouts with kd trees and evolutionary algorithms. In: Generative Art Conf. 2010, p. 238–253.

[17] Koenig, R., Schneider, S.. Hierarchical structuring of layout problems in an interactive evolutionary layout system. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 2012; 26(02):129–142.

[18] Wong, S.S.Y., Chan, K.C.C.. EvoArch: An evolutionary algorithm for architectural layout design. Computer-Aided Design 2009; 41(9):649–667.

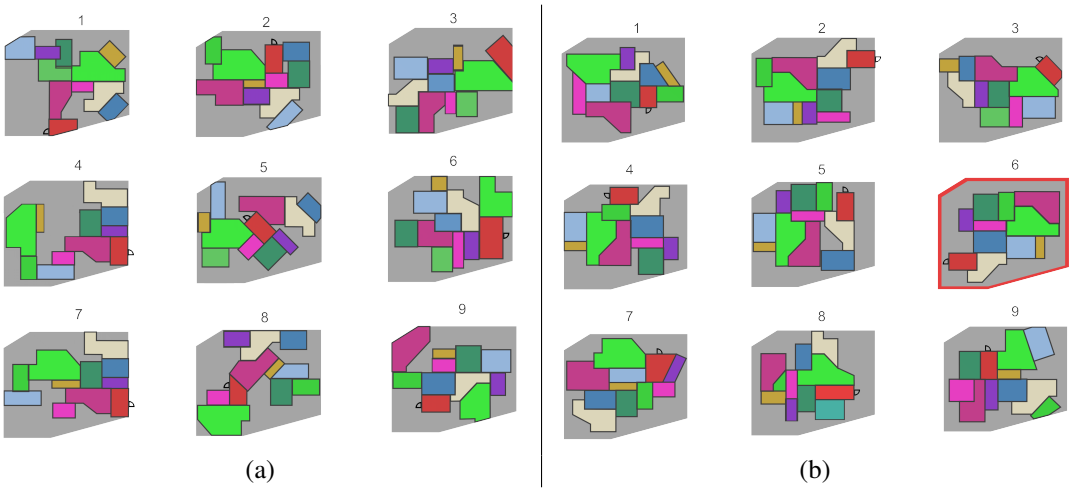


Figure 18: (a) A screen shot of nine layouts of first generation. (b) A screen shot of nine layouts of 86th generation.

Figure 19: Five layouts that were rated by user in different generations.

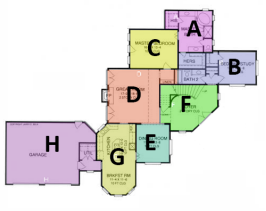


Figure 20: Reference layouts.

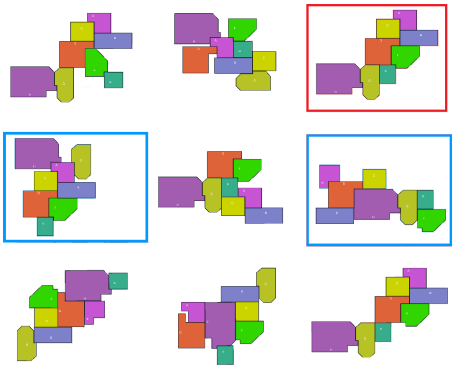


Figure 21: Screenshots of the last generations of the validation experiment

[19] Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.. Procedural modeling of buildings. ACM Transactions On Graphics (TOG) 2006; 25(3):614–623.

[20] Fujiyoshi, K., Murata, H.. Arbitrary convex and concave rectilinear block packing using sequence-pair. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 2000; 19(2):224–233.

[21] Chu, C.C.N., Young, E.F.Y.. Nonrectangular shaping and sizing of soft modules for floorplan-design improvement. IEEE Transactions onComputer-Aided Design of Integrated Circuits and Systems 2004; 23(1):71–79.

[22] Bentley, P.J., Wakefield, J.P.. Conceptual evolutionary design by a genetic algorithm. Engineering design and automation 1997; 3:119–132.

[23] Feng, T., Yu, L.F., Yeung, S.K., Yin, K., Zhou, K.. Crowd-driven mid-scale layout design. ACM Transactions on Graphics (TOG) 2016; 35(4):132,14p.

[24] Cardamone, L., Loiacono, D., Lanzi, P.L.. Interactive evolution for the procedural generation of tracks in a high-end racing game. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM; 2011, p. 395–402.

[25] Bahrehmand, A., Evans, A., Blat, J.. A Computational Metric of the Quality of Circulation in Interior Spaces. In: The International Conference on Information Visualization Theory. Lisbon; 2014, p. 8.

[26] Chen, T.C., Chang, Y.W.. Modern floorplanning based on B*-tree and fast simulated annealing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2006; 25(4):637–650.

[27] Peng, C.H., Yang, Y.L., Bao, F., Fink, D., Yan, D.M., Wonka, P., et al. Computational network design from functional specifications. ACM Transactions on Graphics (TOG) 2016; 35(4):131,12p.

[28] Westin, A.F.. Privacy and freedom. Washington and Lee Law Review 1968; 25(1):166.

[29] Nilgun Kuloglu, T.S.. Perceptual and visual void on the architectural form: Transparency and permeability. Architectonica 2012;:131–137. doi:10.5618/arch.2012.v1.n2.4.

[30] Eiben, A.E., Smith, J.E.. Introduction to evolutionary computing. Springer Science & Business Media; 2003.

[31] Rodrigues, E., Gaspar, A.R., Gomes, Á.. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture Part 2: Validation and performance tests. Computer-Aided Design 2013; 45(5):898–910.

[32] Boddy, M., Dean, T.L.. Deliberation scheduling for problem solving in time-constrained environments. Artificial Intelligence 1994; 67(2):245–285.

[33] house designers., T.. thehousedesigners. 2016.