

Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation

International Journal of
Architectural Computing
2019, Vol. 17(3) 260–283
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1478077119832982
journals.sagepub.com/home/jac


Maciej Nisztuk¹  and Paweł B. Myszkowski²

Abstract

The article presents the application of Hybrid Evolutionary and Greedy-based algorithms to the problem of Automated Floor Plan Generation. The described optimization issue is part of a wider domain of Computer-Aided Architectural Design. The article covers the extensive description of the representation domain model (architectural canonical guidelines, user design requirements and constraints) and the explanation of proposed approach: problem representation, genetic algorithm operators, and fitness function definition. The research experimental procedures are based on real-world data: the architectural design guidelines being the design constraints and five real-world functional programs introduced and proposed as benchmarks. The article summarizes the implementation of the proposed approach, compares the Hybrid Evolutionary Algorithm experimental results with the Greedy-based algorithm, and suggests possible extensions and future research directions.

Keywords

Computer-Aided Architectural Design, optimization in CAAD, Automated Floor Plan Generation, Hybrid Evolutionary Algorithm, optimization, benchmark

Introduction

Computer-Aided Architectural Design (CAAD) applications include a large group of computer software supporting architectural design. This article is an analysis of one of the CAAD issues—an Automated Floor Plan Generation (AFPG). Unlike typical Computer-Aided Design tools, CAAD software is primarily focused on supporting the architectural design process. Since the beginning of CAAD development, tools aiding the architectural design through the creative use of computer computing power began to appear. Research on the problem of architectural design aided by computational tools has been conducted for over 50 years and the early examples can be found in various papers.^{1–3} In recent years, with the growing support of leading

¹Department of History of Architecture, Art and Technology, Faculty of Architecture, Wrocław University of Science and Technology, Wrocław, Poland

²Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology, Wrocław, Poland

Corresponding author:

Maciej Nisztuk, Department of History of Architecture, Art and Technology, Faculty of Architecture, Wrocław University of Science and Technology, ul. Bolesława Prusa 53/55, 50-317 Wrocław, Poland.

Email: maciej.nisztuk@pwr.edu.pl

producers of architectural software, the field undergoes rapid development, with a good example of The Living group⁴ (the Autodesk architectural research team) and their work.^{5–7} The exploration is mainly structured around two trends: enhancement of BIM software capabilities through connection with parametric design platforms (Archicad® and Grasshopper™; Revit® and Dynamo) and particularly promising academic research projects.^{8–10} The growing role of technology in today's world inevitably affects the entire architectural design process. The architectural community seems to recognize the limitations of existing CAAD tools, especially in the aspects as ephemeral as creativity. The development of modern architectural design tools is mainly focused on the later stages of the design process. The architectural commercial aiding software, though very advanced in this role, is developed mainly as advanced “drawing aids.” In recent years, more and more research has focused on supporting the initial, concept stage of architectural design with various approaches showing a great potential in this area, especially for fast exploration of possible architectural design spaces.

Less formally speaking, the AFPG is the computational optimization problem that automates and speeds up the process of searching the architectural solution space for feasible architectural layouts meeting (multiple) design criteria. Generation of a good floor plan layout is quite a difficult process which includes not only a number of measurable and intuitive conditions, but also specific constraints, which must be taken into account in final practical solution/layout. In addition, from the architectural point of view, the AFPG solving method is very useful in practice. The initial stage of each architectural project is creating the concept spatial plan of designed object on the basis of available functional and design guidelines as well as the limitations resulting from the legal constraints. The process of creating the architectural layouts concept is a repetitive search for an optimal solution that meets the criteria and the design constraints. This iterative and analytical process is the arduous search of a wide range of possible design solutions. The computer becomes an useful tool for these purposes because, in contrast to a human, it is adapted to perform many repetitive operations without being susceptible to analytical errors. Authors believe that automation of the search process for the optimum concept of floor layouts may enable

- Broadening of the solutions spectrum meeting the design goals,
- Improvement and acceleration of the architectural design process,
- Architectural design improvement by introducing (multi-criteria) optimization of design solutions,
- Supporting the creativity of designers by generating non-obvious and surprising solutions.

The AFPG problem seems to be currently the main focus with regard to the use and development of computing tools (machine learning, generative design, etc.) in solving architectural problems. The number of requirements and possible combinations of room location on the architectonic floor plan makes this problem extremely complex and computationally demanding, that belongs to the group of nondeterministic polynomial (*NP*) time complete problems.¹¹

Moreover, AFPG defines and allows to effectively search spaces of measurable design conditions leaving architects area for making the most important design decisions. Furthermore, it speeds up the design process by offering solutions meeting (multiple) criteria which can change during design procedure. In order to resolve this problem, hybridized Evolutionary Algorithm (EA) and Greedy Algorithm are applied in this article. This article suggests a new application supporting the conceptual stage of designing architectural floor plans via the AFPG. The application being developed by the authors is called ELISi (EvoLutionary ArchItectural Aided DeSign) and puts the main focus on the usability and functionality achieved by

- The recreation of architectural process stages in the software workflow,
- Intuitive wizard-like interface and workflow (the Graphic User Interface (GUI) is currently at the prototype stage and is described in more detail in the “Future work” section),

- Collection and use of real architectural design data (hereinafter referred to as “Design canons”) to create the correct design solutions,
- Elaborating on the problem based on architectural design practice.

This article focuses on the logical core of the proposed tool which is the Hybrid Evolutionary Algorithm and Greedy-based algorithm constituting the mechanics of generating architectural floors. The authors focus on the conceptual design stage, hence the floor plans are presented in a simplified rectangular form, without architectural openings and other architectural details. It is consistent with the typical architectural design practice, whereas the floor layouts at the early concept stage are presented in the form of rectangular diagrams. Although floor plans generated by the proposed algorithm were created with only single design criterion (evaluation function)—the boundary and layout compactness—their architectural quality is already satisfactory in terms of topological connectivity, room side aspect ratios, and division into functional zones and the room areas to the total floor plan area ratio. Typically, the unused area is a fraction of the floor plan total area which, from the architectural point of view, is negligible at the point of design conception stage. This indicates at the good functioning of the adopted method and its high development potential in planning implementation of the multi-criteria optimization algorithm. A set of architectural tests was developed to examine ELISi efficiency: the benchmark design cases in the form of functional programs of various existing single-family houses—the main focus for the current version of proposed tool. The authors focus on this type of architectural objects due to extensive functional programs in terms of the number of variety of functional connections. Test design cases are used to determine the degree of reliability of the adopted algorithmic solutions with regard to fidelity and similarity of the generated floor plan layouts to the existing real-world examples. It is measured by the degree of similarity between the algorithm-generated plans and the real-world floor plans—the more similarities, the more reliable the method is. It is important to clarify that the use of benchmark data set (consisting of five real-world architectural programs) is designed to verify the correctness and effectiveness of selected methodology, but it does not imply the practicable applicability in the actual design process at this stage. The usefulness of the proposed application will be verified by its future extensive user tests.

The rest of this article is organized as follows. The section “Related work” sets out some of the related works. The “Proposed approach” section presents the AFPG problem definition, the proposed approach, and some implementations details. The provided experiments and results are presented in the section “Experiments and results.” The “Architectural summary of results” section includes architectural summary and comments of the results obtained. The article finishes with the “Conclusion” section presenting the conclusion and the future work.

Related work

The AFPG problem described in this article enables the exploration of architectural design space through the generation of architectural floor plans that meet certain specific design criteria. Similar approaches/problems—a recent survey in Nisztuk and Myszkowski¹²—can be found in the literature.

Merrell et al.⁸ proposes the advanced tool to generate layouts of buildings, taking account of a set of factors such as the functional layout of rooms, area as well as windows and doorways opening. The application provides plausible design outcomes. However, the presented approach uses an idealized situation without site-specific and client-specific features. The prototype generates detailed three-dimensional (3D) architectural models and plausible layout plans. The disadvantage of this solution is the lack of interactive workflow which does not allow to manage the input in real time.

In Willis et al.¹³ is presented an advanced cross-sectional tool for the comprehensive design of one-family houses (catalog designs) used on the US real estate market, created in Grasshopper™. The tool has a complete

approach to house design, from general concept to construction. The implementation of the application with an object-oriented Grasshopper™ visual programming language limits the circle of users to those more experienced. This tool is adapted only to the realities of real estate US domain.

The approach outlined in Bao et al.¹⁴ enables the exploration of potential and accurate building volumes, depending on a set of design guidelines. This tool with a clear graphic interface provides a quick way to create a concept of an accurate building volumes, while the generation process works in real time. Due to a convenient interface, the application is suitable for creating quick concepts of building forms for different urban contexts. The prototype is one of the few convenient and usable tools for the concept stage of architectural design that produces meaningful design propositions.

Described in Liu et al.,¹⁵ the tool allows designing an apartment layout, with a module that generates building structures. Focusing on reinforced concrete constructions, it is one of the few examples of specialized uses of this type of application. Such an interactive tool allows modifying functional properties during the generation process. However, the user can operate only on a single layout of the apartment, and the introduction of more complex functional systems is rather problematic.

A comprehensive tool—GerAPlanO^{9,16}—is one of the most advanced CAAD application and its main purpose is to create a fully functional design tool for architects. The prototype has been created by an interdisciplinary team and in cooperation with architects, development, and a construction company. The intuitive interface enables to use online data (Google Maps, weather data, etc.). It is one of few tools that have a chance to become fully-fledged tools of CAAD using computational design.

The research project that is also worth mentioning is “Project Fractal”¹⁰—led by Autodesk® aimed to explore generative design and optimization techniques of the architectural design process. The tool combines the cloud computing power with Revit® architectural design application via Dynamo scripting environment. It allows exploring the architectural design spaces defined by Dynamo visual programming language. The project is currently in the dynamic development stage becoming one of the first generally available intuitive tools for architectural generative design.

The authors argue that current research, even with the gradual changes toward the end-user (the architects) needs, despite their considerable degree of advancement and satisfactory computational efficiency, is still overly focused on Information Technology (IT) methods resulting in omitting the tool usability aspects. From the point of view of the architect, what is significant is that most of the contemporary applications tend to be created to optimize a partial or the general aspects of the designed structure, rather than enhance the architect’s creativity by providing valid design solutions.

The authors identified and outlined a classification of methods solving floor planning problem. The following list aims to roughly outline computational methods for this problem. A closer analysis and reviews can be found at Nisztuk and Myszkowski.¹² These applications can generally be divided into several groups enclosed in two main categories: with predetermined floor boundary outline, without the specified layout boundary envelope.

The category where predetermined floor boundary outline exists, in general, contains methods for dividing the polygon’s specified boundary contour, based on various computational methods or architectural conditions. The group contains examples like polygonal division—using KD trees,¹⁷ rectangle division based on treemaps¹⁸ or facilitation of Voronoi diagrams in the interior¹⁹ or urban design.²⁰ The approaches from the second group are typically based on the division of the polygonal boundary into smaller fragments based on the importance attributed to given space, for example, based on the number of room connections or area priority.²¹ This category is less useful for computational creation of floor plans. Take the example of the scenario in which one of the main design criteria is the inviolable building outline. It is very uncommon and usually refers to redesigning or an adaptation of existing architectural objects for other purposes or to historic buildings renovation.

The category without the specified layout boundary envelope contains methods for creating floor layouts without a given outline and adjustments for given design requirements fulfillment. Various methods can be distinguished:

- Dynamic addition of rectangular rooms to each other;²²
- Floor layouts creation based on a rectangle divided into multiple fragments corresponding to the number of rooms and subsequent geometric transformations, for example, moving the edges of rooms gradually enlarging the room to the assumed size;⁸
- Modeling by physical simulation, for example, on the basis of the topology graph with nodes depicting the rooms and the edges representing springs. The nodes are attracted to each other with the forces representing given connection preference;²¹
- Floor plans creation based on cellular automata models—the rooms “grow” gradually by adding further small modular blocks, aiming at the given geometrical preferences. This method allows for non-orthogonal space shapes.²³

The above approaches are combined with all different types of solution searching methods, typically based on Genetic Algorithms⁶ or some machine learning algorithms.²³ The authors are aware that the above classification does not fully exhaust the available methods for AFPG problem. However, it roughly outlines the main directions of solving methods.

The conducted literature research allows concluding that the main directions of contemporary tools for the AFPG problem mainly focus on developing an IT method without focusing on the user’s layer. The field lacks a holistic approach to the subject of the AFPG problem related to the development of comprehensive systems that supports the work of architects taking into account the user’s needs in terms of usability and functionality. To address that, the authors introduce the user-centered approach²⁴ which will be elaborated further along with the overall progress on the development of the tool. The proposed approach is signalized throughout the article although it is not specifically described here. The article describes the logical core of the proposed application being the Hybrid Evolutionary Algorithm and Greedy-based algorithm constituting the mechanics of generating architectural floors. The proposed tool aims to gather all good practices and to create a functional architectural software complementing the design work, not necessarily primarily focusing on computational efficiency. It should be noted that, there are other currently developed tools—based on similar assumptions^{6,9,10} which can be perceived as a reflection of a growing user-centered trend in contemporary software development.

Proposed approach

This section sets the main features of the proposed approach—its motivations and description. This article will analyze the current stage of ELISi project. In this section, it will summarize already finished stages and will outline next.

Motivations and previous work

The *novelty* and the main motivation of proposed approach are focused on the recreation of architectural design process stages in the proposed tool workflow and emphasis on software functionality and usability. This can be achieved by

1. Collection of design information through an interactive interface resembling the architectural brief;
2. Intuitive wizard-like application workflow;

3. Conclusion of the user in the evolutionary selection process via Interactive Evolutionary Computation—after each run of the application, the user has the opportunity to select the preferred solution that proceeds to the next pass as the individual with the preferred set of genes;
4. User design requirements and the generated floor plans confronted with the set of actual architectural design constraints (Design Canons).

Before the preparation of this article, some works have been completed as the following stages of ELISi project:

1. Literature studies on available solutions for AFPG such as available algorithmic tactics, architectural design criteria selection, single and multi-criteria optimization methods presented in Nisztuk and Myszkowski;¹²
2. Definition of the main users' software requirements based on a survey conducted on the professional architects presented in Nisztuk and Myszkowski;¹²
3. Development of the application workflow and the software architectural design guidelines presented in Nisztuk and Myszkowski;²⁴
4. The proposals of architectural floor plan notation for AFPG problem: formulation of architectural design criteria set being the selection of criteria for adopted algorithmic method (evaluation function);
5. Collection of architectural design data and the design set development (*Design Canons*) which has been included in the proposed application, being the testing use cases for an algorithmic method;
6. Development of the architectural benchmark set including five examples of architectural functional programs of various real-world single-family houses. This type of architectural object is the main focus for the current tool. It is due to extensive functional (in terms of the number of the variety of functional connections) programs;
7. Development of the AFPG solution method that solves the problem. Proposed approaches use a heuristic (Greedy-like approach) and metaheuristics (like EAs) and hybridize them;
8. Experiments and verification procedure for proposed approaches using five examples of architectural functional programs.

Stages 4–8 are described in this article whereas further ELISi stages are presented in the “Future work” section.

Description of solution model for AFPG

Informal description of the AFPG problem is as follows: for several rooms (defined by a user), a solution is arranging them (placement, location) and fixing their size to optimize their flat bounding box aiming at its smallest area (minimization problem) and meeting the assumed constraints. The AFPG domain constraints should be formulated as follows:

- Set of r_i (where $i : < 1 \dots n >$) of n rooms are given,
- Each room has neighboring rooms defined—a summary of all rooms gives a topography of the analyzed flat,
- For each room r_i are given size (*min* and *max*) constraints $w_i : < w_i^{min}, w_i^{max} >$ and $h_i : < h_i^{min}, h_i^{max} >$, where w_i is a room's width and h_i is room's height,
- For each room r_i are given area (*min* and *max*) constraints $r_i.area^{min} < w_i \times h_i < r_i.area^{max}$ that limit the area of given room.

The solution of AFPG is a *floorPlan*—a sequence of $r_i := \langle x_i, y_i, w_i, h_i \rangle$ that defines r_i area $\langle w_i, h_i \rangle$ and position $\langle x_i, y_i \rangle$ in analyzed flat. Such a problem definition causes a very extensive solution area, where only a small number of solutions comply with the constraints and are applicable. Thus, in the proposed approach solution landscape is reduced only to feasible solutions (all constraints are met). In the selected approach for the **fitness** function, the *bounding box (BB)* value is used and defined as follows

$$fitness_{BB}(floorPlan) = \left(\max_{i=1}^{rooms} (x_i + w_i) - \min_{i=1}^{rooms} x_i \right) \times \left(\max_{i=1}^{rooms} (y_i + h_i) - \min_{i=1}^{rooms} y_i \right) - \sum_{k=1}^{rooms} (w_k \times h_k) \quad (1)$$

This fitness function, for a given *floorPlan* determines the bounding box area size and reduces its value by summed area of all rooms. Such fitness function is minimized, the lower the value, the better. If the value equals to 0, it means that there is no lost space and floor plan is the optimal one. The value is interpreted in cm². The computational complexity of the AFPG solution landscape can be estimated as follows

$$O(n, p) = n! \times p^{4n} \quad (2)$$

where n equals to the number of rooms and p means how “sensitive” should the approach be. It defines the precision of two scaling factors (*height* and *width*) and preferred position of rooms (x_i and y_i). Such approach to work in feasible solution landscape needs to define a construction algorithm. In the proposed approach, Greedy-like algorithm has been defined.

Greedy-like algorithm

In order to operate in the feasible AFPG solution space only, the Greedy-like algorithm has been applied (see Algorithm 1). Proposed Greedy algorithm builds the final floor plan getting only two parameters: the scale factor matrix (one value for each room is defined) and sequence that represents sequence of rooms ordering. The algorithm formally works as follows

Algorithm 1. Greedy-based algorithm specialized to AFPG.

```

1: procedure GREEDY (SCALE(N) SEQ(N)
2:   for room  $\leftarrow 1, N$  do
3:     Flat[room].width* = SCALE[room].width
4:     Flat[room].height* = SCALE[room].height
5:   end for
6:    $X \leftarrow \text{Flat.Width} / 2$ 
7:    $Y \leftarrow \text{Flat.Height} / 2$ 
8:   while |SEQ|  $\geq 1$  do
9:     CurrentRoom  $\leftarrow \text{SEQ}[0]$ 
10:    PlaceRooms(Flat, CurrentRoom, (X, Y))
11:    Neighbors  $\leftarrow \text{CurrentRoom.getNeighbors}(\text{SEQ})$ 
12:    PlaceRooms(Flat, Neighbors, (X, Y))
13:    SEQ := SEQ \ (CurrentRoom  $\cup$  Neighbors)
14:   end while
15:   returns floorPlan(Flat)
16: end procedure

```

$$floorPlan = Greedy(SCALE[N], SEQ[N]) \quad (3)$$

where *SEQ* is a rooms ordering that would be processed by algorithm. The *SCALE* is a set of tuples $\langle w_i^s, h_i^s \rangle$ that defines for every room (separately) the scaling factor. The values of rooms w_i width are modified by $w_i^s \in \langle 0.0, 1.0 \rangle$ scale factor as follows

$$w_i = w_i^{min} + (w_i^{max} - w_i^{min}) \times w_i^s \quad (4)$$

Such operations scale room's width and limit its value according to constraints defined by user in given canon. Analogously, each room's height h_i is modified. The given default values are $w_i^s, h_i^s = 0$.

Developed Greedy-like algorithm (see pseudocode Algorithm 1) after scaling operation (see lines 2–4) initializes the position of first room (lines 6–7) in the middle of the flat layout. Next, the first room in *SEQ* is localized (line 9) and then its neighbors (rooms that have a connection in canon user preferences, see lines 11–12). Then, already placed rooms are removed from *SEQ* queue. However, functionally more important in the presented algorithm is *PlaceRooms* procedure (see pseudocode Algorithm 2). This procedure puts (local) optimal localization for room(s) given as argument in suggested (X, Y) position.

Algorithm 2 uses two architectonic parameters: $offset_{min}$ and $tangent_{min}$, which are given in [cm]. The first value is a compromise that allows to move rooms by 10 [cm] in X or Y axis—the lower value means more computationally complex algorithm, the bigger value may produce more infeasible solution space. The second value ($tangent_{min}$) represents the minimal length of connection edge between rooms that must be adjacent (tangent)— $tangent_{min} \leftarrow 100$ and is interpreted as the 100 [cm] is the minimal space to put door between rooms. This value is consistent with accepted architectural design practice.

The main work of Greedy algorithm is contained in *PlaceRooms* procedure (see Algorithm 2). First, it tries to put given *room* in suggested position (X, Y) (see lines 5–6), then tries to move *room* according to

Algorithm 2. PlaceRooms procedure.

```

1: procedure PLACEROOMS (Flat, Rooms, ( $X, Y$ ))
2:    $offset_{min} \leftarrow 10$ 
3:    $tangent_{min} \leftarrow 100$ 
4:   for room  $\leftarrow 1, Rooms$  do
5:     Flat.locateRoom(room, ( $X, Y$ ))
6:      $Eval_{best} \leftarrow Flat.eval(room, (X, Y))$ 
7:     for side  $\leftarrow 1, room.sides$  do
8:       while Flat.isNeighborsTangent(room,  $tangent_{min}$ ) do
9:         ( $X, Y$ )  $\leftarrow offset(offset_{min})$ 
10:        if Flat.noCollisions(room) then
11:           $Eval = Flat.eval(room, (X, Y))$ 
12:          if  $Eval < Eval_{best}$  then
13:            ( $X, Y$ )best  $= (X, Y)$ 
14:             $Eval_{best} \leftarrow Eval$ 
15:          end if
16:        end if
17:      end while
18:    end for
19:    Flat.locateRoom(room, ( $X, Y$ )best)
20:  end for
21:  returns ( $X, Y$ )best
22: end procedure

```

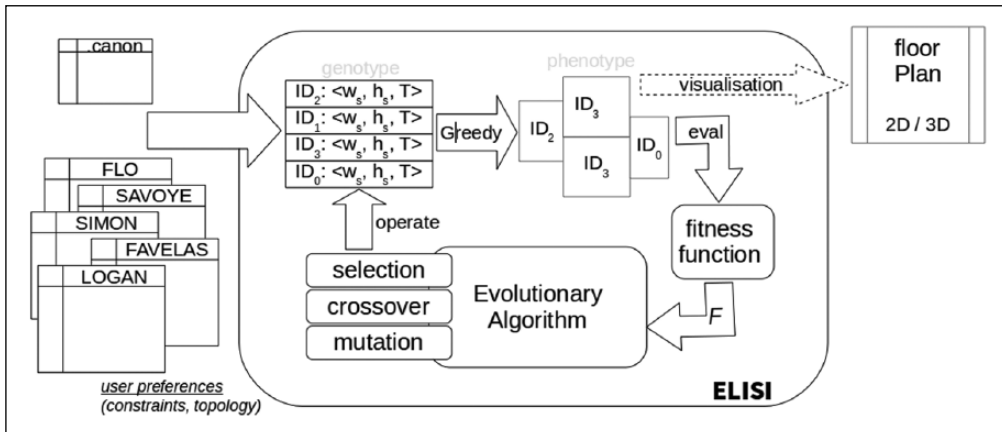


Figure 1. General schema of ELISi model.

its four sides using minimal offset $offset_{min}$ value. Such local search approach is realized if the room is tangent to its neighbors (see line 8) and does not make any collisions (two or more rooms are overlapping, see line 10). The already found position is marked (lines 12–14) and finally, the room gets the best position found (see line 19).

The preliminary research results showed that Greedy-like algorithm strongly depends on the initial sequence of rooms and initialization method (heuristic or a random one). In literature, hybrid of Evolutionary Computations and Greedy-like algorithm is known as very effective. This is the reason why approach presented in this article is slightly inspired by the idea of hybrid DEGR (Differential Evolution and Greedy²⁵). However, this article uses EA, whereas Greedy-like is specialized to solve AFPG problems. In such hybrid, EA works to find the optimal initial sequence of Greedy-like and also proposes a room scaling aspect to optimize flat area.

Hybrid Evolutionary Algorithm and Greedy-like algorithm

A Greedy-like algorithm constructs the final floor plan and as an input it takes *Greedy(SCALE, SEQ)* sequence of rooms and scaling values for each room. The potential landscape of all inputs encourages authors to apply EA to find the potential optimal input for Greedy-like algorithm. Such approach is Hybrid Evolutionary Algorithm and Greedy-like algorithm—the whole system is called ELISi (EvoLutionary ArchItectural Aided DeSign). Schematically, it is presented in Figure 1.

In the ELISi data flow (see Figure 1), the constraints given by user are the most important. The requirements include a number of rooms, types of room, rooms topology, and minimal/maximal length/width/area of a given room. If any constraint is not given by the user, its value is copied from predefined architecture canon file. In the next stage, EA is initialized and genotypes are randomly filled. Then, Greedy-like algorithm is executed to transform genotype into phenotype (floor plan) to be evaluated by fitness functions. Evaluated genotypes are then selected, modified by genetic operators (crossover and mutation) to create a new population. The evolution cycle is closed until it does not meet one of the stop conditions: a number of generations exceeds the limit or individual with fitness function equal to 0 is found. The best individual is selected as the final 2D/3D floor plan and is presented to the user.

The ELISi solution representation (genotype) consists $\langle SCALE[n], SEQ[n], T[n] \rangle$, where $SEQ[n]$ defines sequence of rooms and $SCALE[n]$ defines for each room $\langle w_i^s, h_i^s \rangle$ scaling factors. Additional

matrix $T[n]$ defines for each room if it should be transformed $t_i \in \{false, true\}$ and rooms axis should be changed. For the above representation, ELISi uses three types of mutations: (1) room sequence swap, (2) Gaussian modification of room scaling $\langle w_i^s, h_i^s \rangle$, and (3) transformation mutation $t \in \{false, true\}$. The swap operator changes the ordering of two random rooms in chromosome according to P_m probability. The second operator allows tuning chromosome by small changes by the Gaussian distribution mutation. The last (transformation) mutation changes room's axis $w_i^s \Leftrightarrow h_i^i$. This transformation gives the additional diversity of solutions and improves evolution.

As a crossover operator, the standard SX (*Single-Point Crossover*) with probability P_x which swaps parts of two chromosomes and produces two offsprings is applied. The EA includes random initialization and tournament selection method. Finally, ELISi uses five parameters: population size pop_{size} , number of *generations*, mutation probability P_m (the same value P_m is used for all mutations), crossover probability P_x , and tournament $tour_{size}$ selection size.

Experiments and results

The aim of the conducted experiments was to investigate the robustness and efficiency of ELISi approach in comparison to results obtained by other methods: Greedy-based variants and multiStart Greedy. The obtained results are described by fitness value [cm^2] and computational time using five developed architectural cases studies.

Architectural case studies and design canons used in the study

To test the developed ELISi, the authors created the benchmark data set consisting of five real-world architectural programs along with their simplified geometrical (rectilinear) representations. The benchmark set has been prepared for the broadest possible diversity of functional systems and design conditions. The collection contains three various contemporary programs (Flo,²⁶ Logan,²⁷ and Simon²⁸) representing the archetypal functional systems of a single detached house, one example of a classical modernist architectural layout (villa Savoye²⁹) with extensive functional program, and one example of vernacular and organic program of house located in Rio de Janeiro's dense urban favelas area.³⁰

*Flo III GI*²⁶ is a popular catalog project of a detached single-family home designed for a wide range of consumers provided by the courtesy of Archipelag design studio. One story house for three to four people, without basement, with attic for residential adaptation and garage for one car. The building includes three bedrooms, a spacious living room connected directly to the terrace, and a kitchen. The design is maintained in modern style preserving the archetypal symbolism of the house, with clearly separated living, private, and utility spaces zones.

*Villa Savoye*²⁹ is a modernist villa built between 1928 and 1931 for the Savoye family, located in Poissy, in the suburbs of Paris, France, by Swiss architect Le Corbusier and Pierre Jeanneret. It is a compact, three-story detached building on a rectangular plan, located on a meadow surrounded by forest. The ground floor consists mainly of guest rooms, a spacious hall, and a garage. The first floor is the main living space with kitchen, living room, and spacious rooms. The second floor contains roof terrace and sculptural structures. Vertical communication is provided by main communication ramp with additional staircase. The building is an icon of modernist architecture thanks to its minimalist, open space, large windows, and the interconnection of interior and exterior space. The current implementation takes into account only ground floor functional program.

*Logan GI*²⁷ is an example of popular catalog detached single-family home designed for three to five persons provided by the courtesy of Archipelag design studio. The house has a traditional layout with the clear division for day zone located on ground floor and night zone located on the first floor. It is a two-story

house with an attic, without a basement and with garage for one car. The design is maintained in modern style preserving the archetypal symbolism of the house, with clearly separated living, private, and utility spaces zones. Prepared project includes basic information about the architectural program and layout geometrical properties.

*Simon G2*²⁸ is another example of a popular catalog single-family house design provided by the courtesy of Archipelag design studio. The building is maintained in modern style preserving the archetypal symbolism of the house. It is a detached, one-story house, with attic for later adaptation to residential purposes, without a cellar, with garage for two cars, designed for a family of four. It contains a clearly separated daily area consisting of a living room connected to a kitchen and a terrace and private zone with three independent bedrooms. In the entrance area located at the front of the building, the utility space was separated along with a two-bay garage. Developed project includes basic information about the architectural program and layout geometrical properties.

*The favelas*³⁰ is an example of typical single-family, three-story building located in the Complexo do Alemão neighborhood, one of Rio de Janeiro's dense chaotic urban favelas area. The building is a peculiar example of vernacular architecture. Despite the lack of the architect's involvement in the design and construction of the object, it retains orthogonal geometry and classical zoning division. The ground floor is occupied by living (day) zone (kitchen, living room, bathroom, and dining room), bedrooms (night zone) are located on the first floor, the top floor (roof) is a multi-purpose terrace. Current benchmark counts only the ground floor. The project includes basic information about the architectural program and layout geometrical properties.

The user can propose his or her own design constraints by changing the default design canon reference database; the design canon contains a set of default design information about dimensions, area, and connectivity patterns for each room type occurring in a functional program of a typical single-family home. Such information is the "canonical" design constraints for the generated floor plans. Thus, if the user does not provide certain functional information required for the layout, for example, the preferred area of room type included in the desired functional program, the program supplements this information from the design canon file. The canon data were determined in accordance with the architectural design practice and common architectural knowledge contained in the architect's design manuals such as Neufert.³¹

In addition, in the provided canon, each room type contains information on its minimum and maximum length, width, and area. Furthermore, the data on preferred floor and location relative to the directions of the world are present. Also, favored topological connectivity between the specified room types is present in the form of the adjacency matrix. The design canon includes instructions on how areas of specific room types should be treated during the generation process as well. For example, typically the communication and utility spaces should have the smallest area possible contrary to the larger area of daily and night zones. Based on architectural design practice, some rooms do not contain data for maximum value of the area, length, or width, for example, living room area is usually determined at the functional program stage and is referred as a percentage of the total usable area.

All presented results (floor plans), five case studies and canon details are developed by *xml* format files and available online (<https://www.ii.pwr.edu.pl/~myszkowski/caad/>). The input file contains constraints and required topology and uses self-describing XML format, for example, it gives the root element that contains successive rooms descriptions whose main elements are *id*, *category*, *dimensions*, and *neighbourhood*. There is also *floor*, *preferredDirections*, and *areaTendency* to describe other aspects of room elements. Field *Dimensions* contains *dim* elements. Each element holds single value which describes requirement specified through name, for example, *minWidth* is a minimum allowable value of width for given room. There are also measurement and unit elements for future use. The room's ranges (*min,max*) can be specified for such dimensions: *width*, *length*, and *area*.

Table 1. Summary and comparison of ELISi results (given in cm²).

Canon	Greedy (R, 10k)		Greedy (D)	Greedy (S)	ELISi		
	Best	Avg	–	–	Avg	SD	t [s]
Favelas	41,400	154,372	233,000	264,400	11,290	3668	306
Flo	139,800	428,196	394,800	520,700	34,307	28,273	4436
Logan	161,200	369,758	477,300	351,600	65,433	28,814	3299
Savoie	130,300	361,105	560,600	519,800	24,563	20,529	4316
Simon	208,800	515,413	675,400	840,100	35,383	31,012	6510

Experiments' procedure

To investigate ELISi performance, some experimental procedure is needed. The ELISi is implemented using python language with standard libraries. As proposed approach is based on EAs that consists of some non-determinism elements, all empirical experiments of ELISi were repeated at least 30 times and results are averaged. All experiments are realized on the computer configured CPU i7-2630 200 GHz, 16 GB RAM (Windows 7 OS).

The basic method for comparing ELISi results is Greedy-like algorithm that uses two types of initialization heuristic: (D)egree that orders descending all rooms by the number of connections and (S)ize that orders descending all rooms by its area. As the additional benchmark, the (R)andom initialization is also used. The procedure is repeated 10.000 times—results are averaged and best value is also given.

ELISi has been tuned experimentally. The best found ELISi configuration is $pop_{size} = 100$, $generations = 100$, tournament selection $tour_{size} = 10$, crossover probability $P_x = 0.1$, and mutation probability $P_m = 0.015$.

Results

The goal of the conducted experiments was to investigate the robustness and efficiency of ELISi in comparison to results obtained by Greedy-like variants. The obtained results are described by fitness values and computational time needed to get a solution. Summary results are presented in Table 1. There are results of Greedy-like algorithm that uses three variants of initialisation (R)andom order, room (D)egree order, and room (S)ize order given. Two initialization methods (D and S) are deterministic and their outcome is always the same result. Thus, (R)andom initialization is repeated 10.000 times and is averaged. As a result, multiStart Greedy method is investigated and results could be easily compared to ELISi results that uses 10.000 births (100 individuals that evolved by 100 generations).

Results presented in Table 1 show that Greedy-like algorithm heuristic-initialized by room (D)egree and (S)ize is deterministic but not effective. The results of (R)andom initialization of Greedy-like algorithm are more interesting: it gives on average 21.42% lower fitness than (D)egree initialisation and 24.68% compared to (S)ize initialisation results.

In order to compare multiStart Greedy-like algorithm and ELISi result, the best result of Greedy and averaged results of 30 runs of ELISi are given. Results contained in Table 1 show that for each case, ELISi is more effective and returns better solution—in average solution is 74.36% better than multiStart Greedy. The average computational time to get the solution is 3773 s and this value can be easily reduced by using more powerful computer, non-scripting programming language (like C/C++) and more restricting programming techniques. In this stage of experiments, however, such work is not considered.

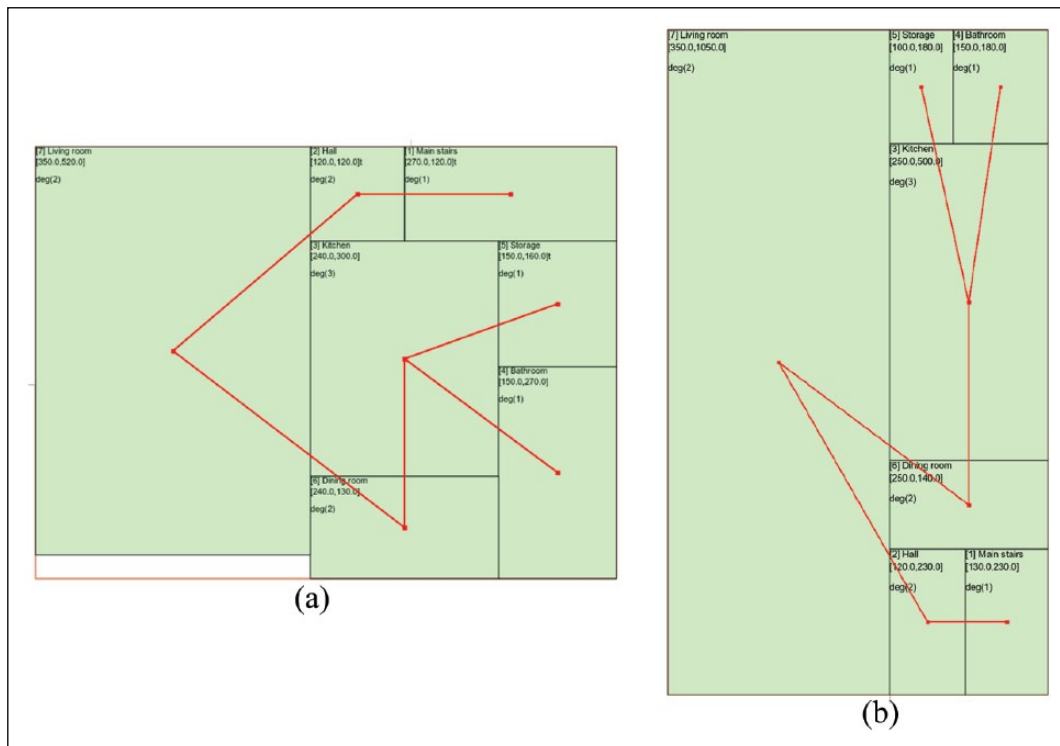


Figure 2. Favelas: selected examples of solutions (a) near optimal ($\text{fitness}(x) = 10,500$) and (b) optimal one ($\text{fitness}(x) = 0$).

Data presented in Table 1 show another potential further directions of research—specialization of genetic operators. The current ELISi version uses only standard operators—this may cause that ELISi can stuck in local optima, which suggest a relatively large values of standard deviation of gained results (see in Table 1 SD column). Application of specialized genetic operators makes evolution more effective, allows avoiding sticking in local optima and gives more repeatable results.

Architectural summary of results

The authors chose 11 sample results achieved with the current version of ELISi. Two for each benchmark floor layouts and an additional example of *Simon* result with fitness function equal to 0 (the optimal one). Each result is identified by the code run number and the name of the benchmark. Some close to optimum results are presented as ELISi example solution. Such examples are worth analyzing because they typically need only small improvement to achieve the optimal scores. Figure 2(a) (10,500 cm² is unused) can be used as an illustration—the living room can be extended by empty area and nearly optimal solution is gained. The unused area (10,500 cm², or 1 m²) is a fraction of the floor plan total area. From the architectural point of view, especially at the concept design stage, this is not a major problem.

It is important to mention that the only optimization criterion included in this research is the minimization of the layout total area in relation to *boundary box* area condition and the topological connectivity of spaces. No room location related to world sides, no room area criterion, or proportions of the sides of the rooms, no

other important optimization criterion is present. Despite this, the system already managed to create functional architectural layouts with rooms of good proportions and with clear zonal separation. The rooms requiring bigger area (living room, garage) are larger, the spaces which area should be minimized (toilets, communication, storage units) are clearly smaller and grouped. Only the ground floor of the benchmark is currently taken into account. Because of that, some spaces are not present on the generated floor plans. *Logan* and the *Favelas* lack the sleeping zones (located originally on the first floor) while *Savoye* lacks the day zone (living room, dining room, kitchen, etc.).

Design canon: Favelas

The *Favelas* contains seven rooms and is the least complex functional program. This benchmark example is described by the non-standard vernacular functional program being a response to the dense urban favela neighborhood, thereby can violate general rules of architectural design. Figure 2 presents two selected solutions of favelas design.

Figure 2(a) presents a design where the kitchen, storage, and bathroom are grouped together, creating a closed utility zone, easily accessible from the day zone (living room and the dining room) resembling the original benchmark solution. The main entrance zone (the hall) is grouped together with the main stairs creating the clear communication zone. The kitchen is located in the central part of the floor layout, lacking the daylight, which agrees with the functional program of the original floor plan.

An example of Favelas solution (see Figure 2(b)) achieved the 0 fitness score (optimal one), creating the compact floor layout. The functional division of the original is in general preserved with the main entrance located in the hall, although the spaces are in different positions and the living room area is large in relation to total floor plan area. In addition, the resulting rooms lack the correct proportions and are, in general, too narrow.

Design canon: Logan

This example contains 10 rooms. Two examples of Logan project solutions are presented in Figure 3. Figure 3(a) presents a design where the floor plan resembles the original benchmark in the general functional and zonal division. The main entrance is located on the porch. The day zone (living room and kitchen area) is located at the top of the floor layout, while the utility zone with the studio is located centrally with the double garage attached at the bottom. The rooms have good size proportions and areas in relation to the total layout area.

Figure 3(b) presents another example of Logan solution. The floor plan resembles the mirror image of the original benchmark in, generally, functional and zonal division. The main entrance is located in the porch room. The day zone (living room and kitchen area) is located at the bottom of the floor layout, while the utility zone with the studio is located centrally with the double garage attached at the top. The rooms have good size proportions and areas in relation to the total layout area. There is a gap inside the floor, near the cloakroom, which is the disadvantage of this example.

Design canon: Flo

This example contains 11 rooms. The result presented in Figure 4(a) clearly resembles the original benchmark layout (rotated by 180°) containing all the main functional features. The main entrance is located in the porch. The example contains minor differences like the bathroom and living room different positions in relation to original benchmark but still remain in the associated functional zones. The rooms lack the proper proportions and are in general too narrow.

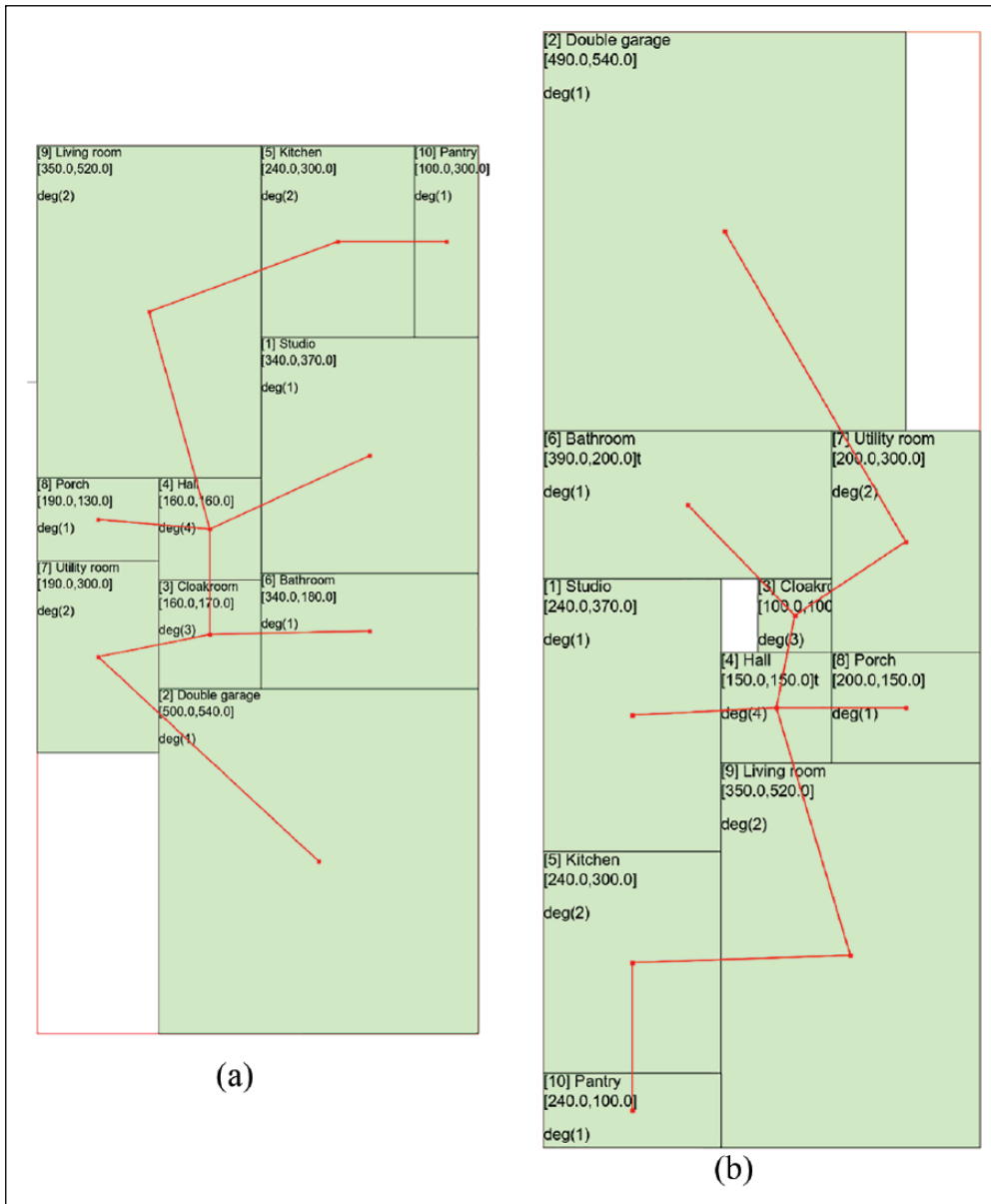


Figure 3. Logan: selected examples of two near optimal solutions. (a) fitness(x) = 83,600 and (b) fitness(x) = 59,000.

The example of the Flo benchmark program presented in Figure 4(b) achieved 0 value of fitness function, creating the compact rectangular floor plan. The main entrance is located on the porch. The bedrooms are grouped together creating clear night zone separated by the centrally located hall. The day zone (kitchen and living room) are divided from the rest of the house, creating the day zone, although the kitchen area is too big in relation to living room space. The single garage with the current dimensions should be rotated in order

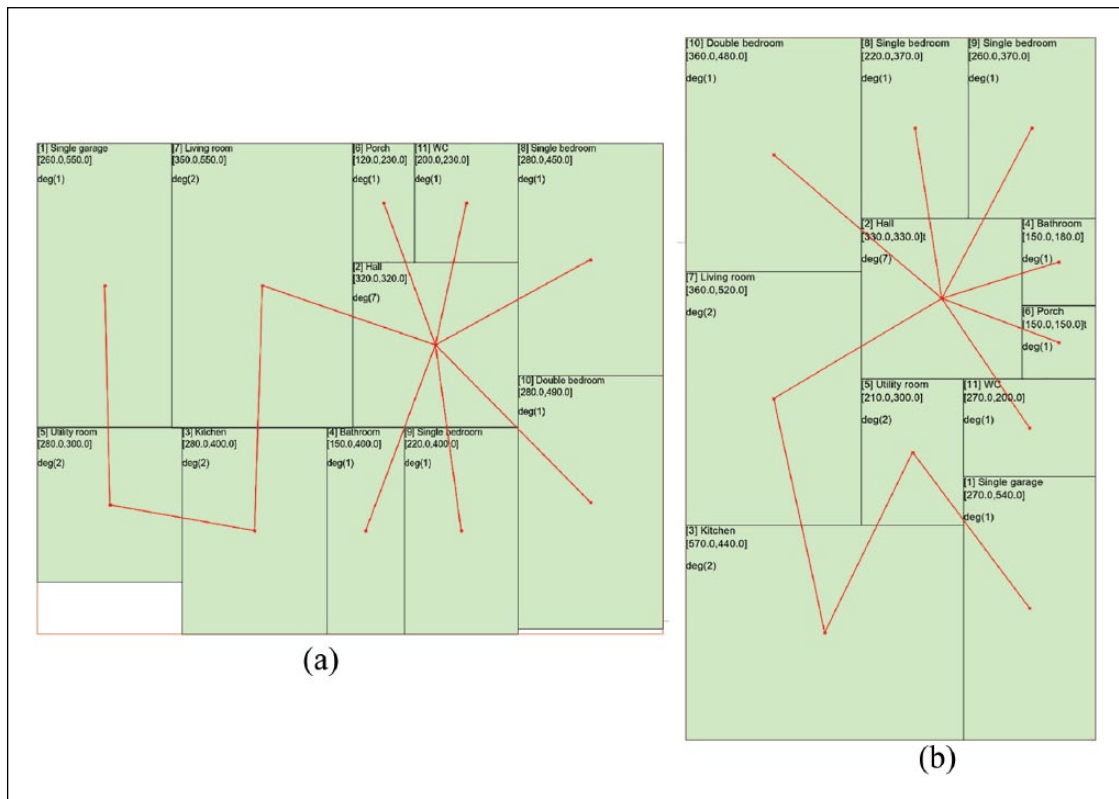


Figure 4. Flo: selected examples of solutions: (a) near optimal (fitness(x)=30,800) and (b) optimal one (fitness(x)=0).

to locate entrance at the side of the main entrance to the house. The rooms have generally correct proportions although spaces 4 and 10 can be perceived as too narrow.

Design canon: Simon

The *Simon* contains 13 rooms, and is the most computationally complex functional program.

The example of *Simon* solution presented in Figure 5(a) shows that the main entrance is located on the porch. The utility room, bathroom, and WC are grouped together in the central part of the layout, creating a closed utility zone, easily accessible from the main communication space and the garage. In addition, the bathroom and WC are close together allowing the join plumbing installation. The bedrooms are grouped together creating clear night zone separated by the centrally located hall. The day zone (kitchen and living room) are visibly divided from the rest of the house. In general, the spaces have good proportions, although pantry and the cloakroom are too narrow creating rather useless spaces. The floor plan contains two inner undesirable gaps.

Another example (see Figure 5(b)) even though with the best possible fitness score did not satisfy the basic architectural condition—the entrance to the building is inside the building. The next example (see Figure 5(c)) defines the main entrance in the porch. The cloakroom, utility room, bathroom, and toilet are grouped together in the central part of the layout, creating a closed utility zone, easily accessible from the

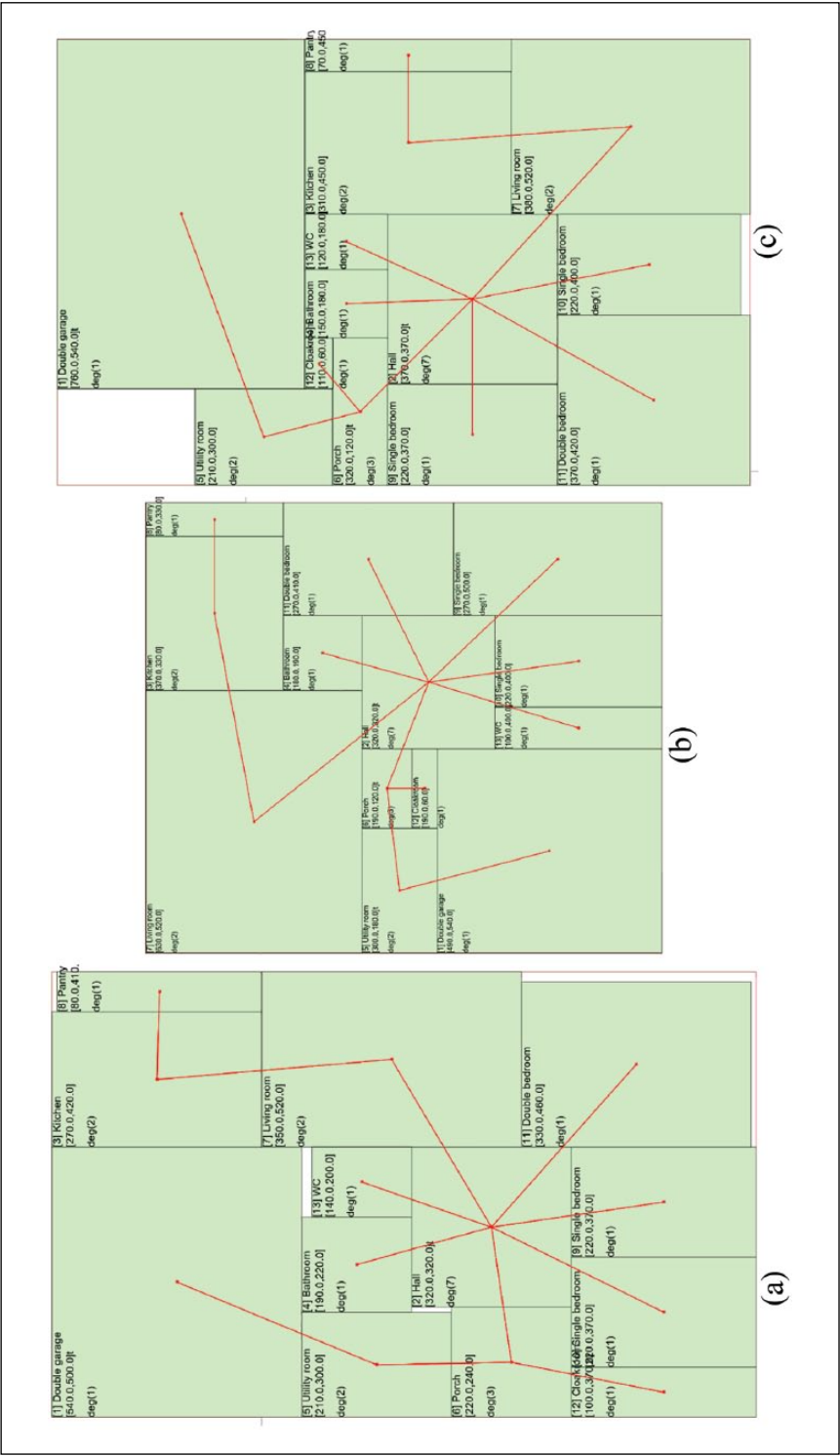


Figure 5. Simon: selected examples of solutions: (a) near optimal ($\text{fitness}(x) = 17,100$), (b) the optimal one ($\text{fitness}(x) = 0$), and (c) non-optimal ($\text{fitness}(x) = 67,400$).

kitchen and from the garage. In addition, the bathroom and WC are close together allowing the join plumbing installation. The bedrooms are grouped together creating clear night zone separated by the centrally located hall. The day zone (kitchen and living room) are visibly divided from the rest of the house, creating an intimate day zone. The pantry is too narrow creating rather non-functional space. The cloakroom is slightly too small to handle the needs of the entire layout. The entrance to double garage is blocked by the utility room location.

Design canon: Savoye

The *Savoye* program contains 12 rooms and is the most functionally complex, most elaborated architectural program containing the non-standard architectural solutions—such as a ramp for vertical communication, the location of the day zone on the first floor, floor plan shape resulting from a combination of a rectangle and a circle. The complication of the floor plan caused in the obtained results (although in general architecturally functional) does not resemble the original and contains strange design solutions (no clearly defined entrance zone, the vast hall space in the benchmark layout, no stairs inside the hall space). The current implementation does not allow creating indoor rooms. Lack of area criteria and the location of rooms relative to the world sides resulted in a different arrangement of rooms relative to the original floor plan.

An example of the solution presented in Figure 6(a) is centrally located communication space (hall, corridor, and the main stairs grouped together) which, in general, divides the night zone from the double garage. Single bedroom (room no 6) has incorrect proportions being too narrow. The original main entrance space (hall) has a central inner location; however, it is connected with the outer boundary by the corridor space. This allows to assume the main building entrance being placed inside the corridor location.

Another solution example (see Figure 6(b)) defines the centrally located communication space (hall, corridor, and the main stairs grouped together), which, in general, divides the night zone from the double garage. Although the original main entrance space (hall) has inner location, it is connected with the building boundary by the corridor space. This allows placing the main building entrance in the corridor space.

Summary and comments

The presented approach is effective and returns a useful application for architect to solve AFPG problem. Using evaluation function and Greedy-like algorithm makes it possible to find a (near) optimal solution. Moreover, the near optimal solution can be further improved by additional algorithmic procedure to extend the rooms near to empty areas.

However, one can be concerned that the current floor plan description is too simple—it is based on the rectangular room outlines, there is one evaluation function (the layout compactness criterion) bounding the floor plan boundary to the rectangle shape. The authors focus on the conceptual design stage, hence the floor plans are presented in a simplified form. It is consistent with the typical architectural design practice in which the floor layouts at the early concept stage are presented in the form of rectangular diagrams. Although relatively simple description, the method proved its effectiveness and great development potential. Presented results are architecturally consistent resembling the original floor plans of benchmark functional programs—both geometrically and functionally. The unused area typically constitutes a fraction of the floor plan total area—which, from the architectural point of view, at the concept design stage is not a major issue.

Conclusion

The presented ELISi application is effective and gives a useful application for the architect to solve AFPG problem. The ELISi results show that AFPG problem formulation and developed method can be effective in

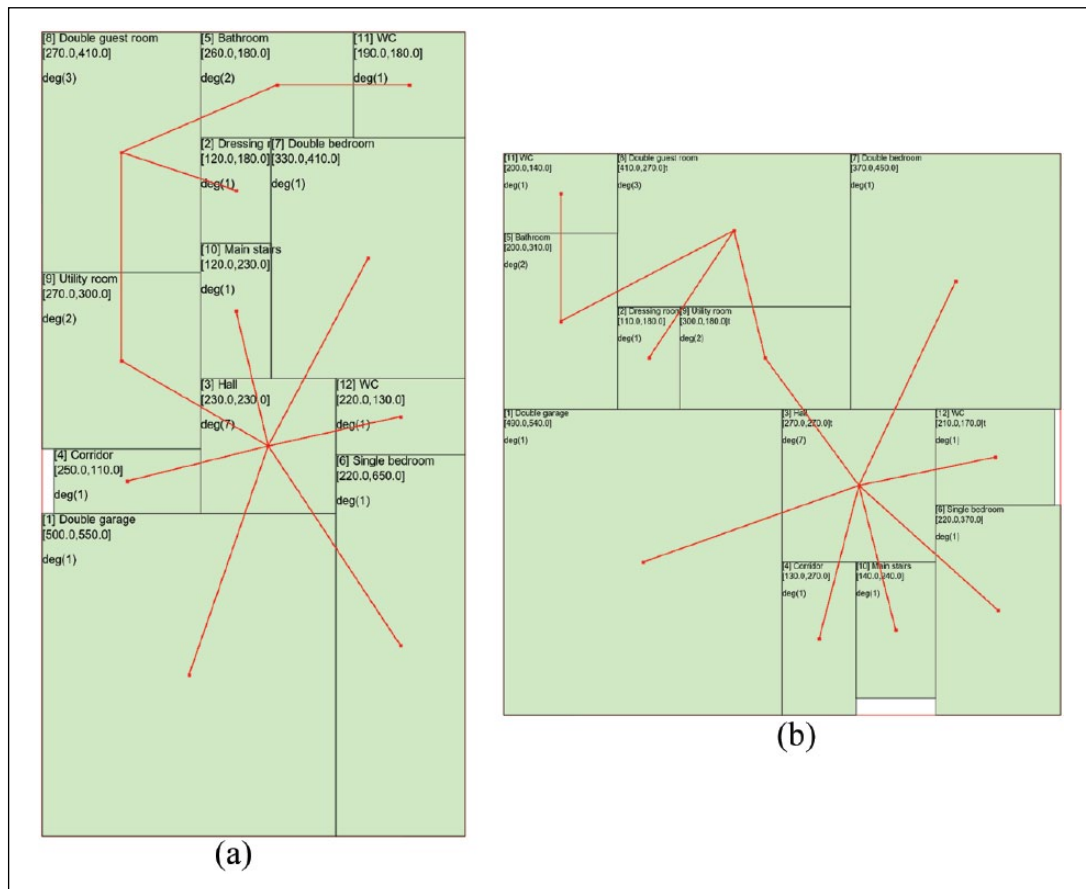


Figure 6. Savoye: two selected very near to optimal solutions. (a) fitness(x) = 2200 and (b) fitness(x) = 5900.

practical design—five architectural cases are solved and analyzed carefully. Moreover, in this stage, ELISI contains some very important properties. Modularity of application implementation enables simplicity of application development for additional evaluation criteria, functionalities, and addition of other design canons. In addition, the implementation model allows development for various platforms, especially as a web platform which gives the possibility of cloud computing and allows the use by the wide audience. The flexibility of implementation allows for application scalability. For canon criteria specified in the current implementation (room connectivity, room dimensions, and area), the architectural design outcomes are very satisfactory. Finally, the application focuses mainly on a conceptual stage of the architectural design process enabling to explore plausible architectural layouts.

Currently, application does not support multi-floor functional programs. This functionality will be added in the future versions through additional room connectivity parameters. Furthermore, the application does not yet contain room dimension aspect ratio evaluator which sometimes results in disproportionate spaces. The architectural design outcomes of the tool are not fully satisfactory. The main reason for this is that the current implementation takes into account only three evaluation criteria regarding the room connectivity (topology), basic dimensions, and area. Additional assessment functions included in further development—namely, the compactness evaluator, the room location relative to the world sides evaluator, and the room

aspect ratio evaluator—should improve even more significantly (already satisfactory in many aspects) the quality of the architectural layouts. In addition, ELISi results show that relatively long computational time could be optimized (e.g. using programming languages more effective than python). Moreover, specialized genetic operator could be very useful.

Future work

The proposed ELISi tool, in this stage of research, is rather a prototype that is focused mainly on generating meaningful architectural floor layouts and the development of fast, architecturally reliable and computationally efficient methodology for AFPG, currently resulting in the limitation of the interactivity with the user. Currently, the user has no influence on the evolution directions by intermediate stages where he or she could participate in the selection process. The current implementation has no integration with existing architectural design software. Further development will be focused on the creation of Grasshopper plug-in which will allow the direct integration with leading architectural BIM software (Archicad).

Once the current application becomes fully functional and optimized (e.g. by the introduction of *multi-threading* computation), the authors are planning to implement several further improvements and new functionalities. The authors are primarily planning to introduce an additional floor plan repair post-processing stage to clear unwanted holes between rooms. To achieve this, the authors assume the algorithm assigning free spaces to adjacent rooms by the area or topology priority. Also, the authors are planning to introduce multi-objective optimization including the room preferred areas, room location relatively to the world sides, and room inner/outer location. The future works also include the introduction of floor levels, architectural openings (door and windows), and the GUI enabling the use of this tool by non-advanced users.

The GUI prototype is designed to allow easy work with ELISi software to any (even non-advanced) user. The main users of this proposed tool are the architects. In order to facilitate the input data collection (the design requirements), the GUI introduces the concept of wizard-like interface and workflow resembling the completion of an architectural design brief familiar to every architect. The interface is created to enable the efficient and intuitive work with the software allowing to focus on the most important aspects of the architectural design process.

The functional extension of ELISi should not only contain integration to some standard architectural application but also EA improvement: greedy initialization, specialized genetic operators, and/or specialized representation. Moreover, the fitness function can be extended to make a solution more practical for architectural usage, for example, room proportion size, room type preferences, and cardinal directions orientation. Another direction is the complexity of fitness function—multiple objectives make AFPG problem the subject of *multi-objective optimization*. The metaheuristics solving this type of problem should use knowledge about domination relation and gained Pareto Fronts like classic NSGA-II³² or its recent modification NTGA.^{33,34}

The multi-objective optimization uses more than one objective function. The most intuitive candidates for objective function are

- *Bounding box*—already implemented and examined in solving AFPG as one criteria optimization;
- *Room preferred areas*—the preferred room area evaluation criterion is a sum of differences between the preferred area for a given room specified by the user (measured in cm²) and the current area of this room. The sum value can be minimized;
- *World sides location* for rooms—each world side can be assigned an angle value that links room orientation and world side. Such fitness function allows to define preferences for world side orientation of given room (e.g. living room located on the south side of the building).

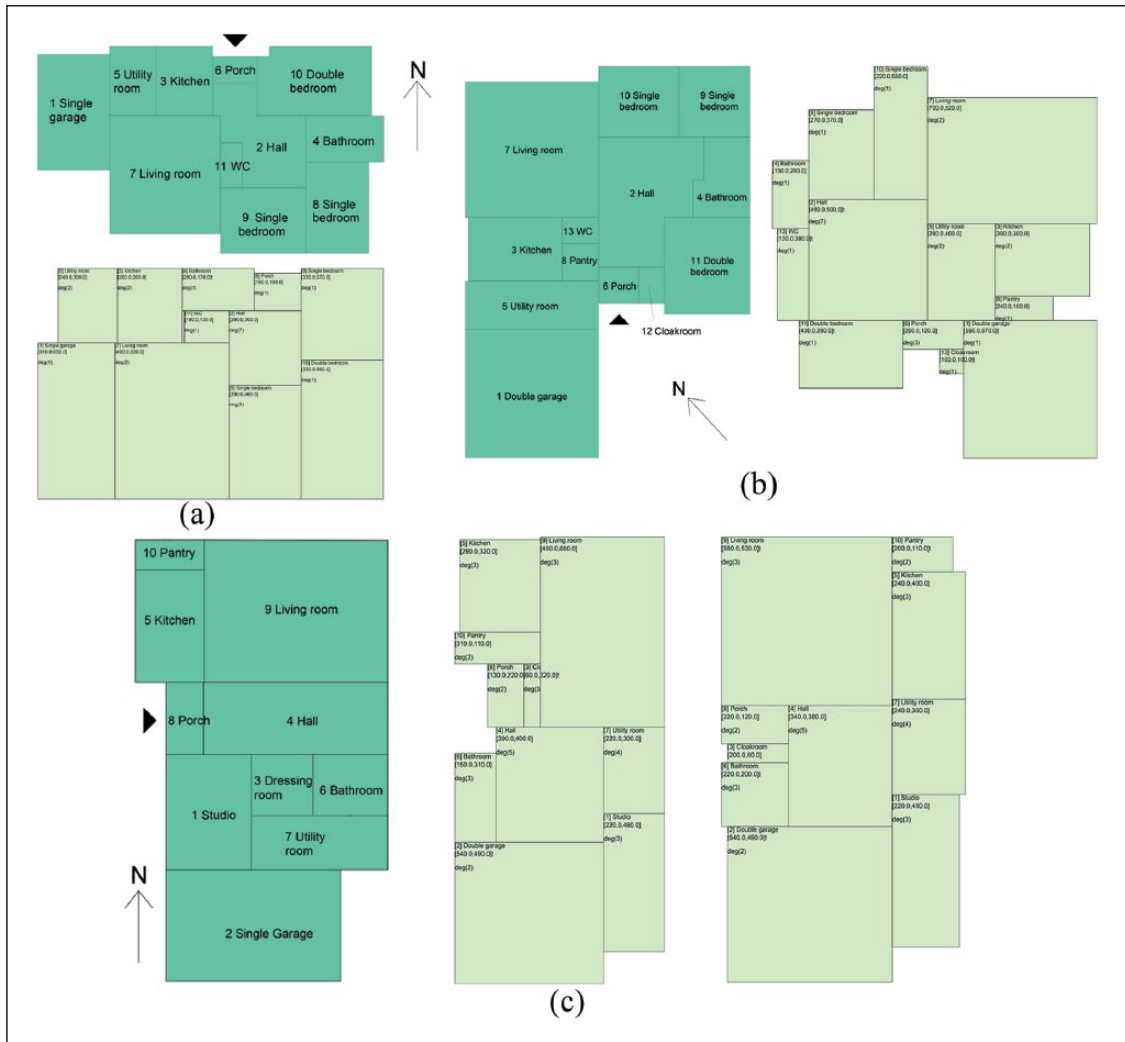


Figure 7. Comparison: of the benchmark original floor plans (dark green) and the solutions generated by the program (light green) after implementation of additional evaluation criteria: (a) Flo example, (b) Simon example, and (c) Logan example.

To show potential and usability of additional fitness functions, let's consider three examples that use the additional evaluation criteria implementation to the current mechanics of floor plan generation. The additional evaluation criteria are room preferred areas and the location of rooms relatively to world sides. In the current examples, the existing fitness function was extended to the form of simple sum of partial values responsible for the given evaluation criterion and it is minimized. It is a simple approach which is not a multi-objective optimization that uses dominance relation. The resulting floor layouts almost perfectly match their real-world counterparts, maintaining the topographic relations of the rooms, their original area, and location relative to the world sides. The illustration (see Figure 7) shows a comparison of the benchmark

original floor plans (their simplified rectilinear representation—dark green) with the solutions generated by the program (light green). Particularly noteworthy is the example of Simon (see Figure 7(b)) where the result is a mirror image of the original. Figure 7(a) shows the original Flo benchmark floor layout and the example created by ELISi. Both layouts overlap in the general arrangement of functional zones, room areas, and topological connectivity. On the right side of the floor layout, there is a sleeping area consisting of three bedrooms and a bathroom, separated from the living area (living room, kitchen) and utility zone (utility room and garage) by a central communication zone. The resulting layout is compact. Small differences in the location of rooms (bathroom, double bedroom) maintain the consistency of functional zones. Figure 7(b) presents an example of the Simon benchmark (the most complex functional system). The layout created by ELISi is a mirror image of the original. Functional zones and the room areas coincide with the original. The resulting floor does not have a regular envelope as the original but retains its general geometric and topological properties: an extended garage located near the utility and kitchen zone connected with the living room separated from the sleeping area by a communication zone. Figure 7(c) shows examples of Logan benchmark floor layouts created by ELISi. Both generated layouts are compact, the location of zones and the main spaces (kitchen, living room, garage) coincide with the original. The differences in the central part of the layout occur within one functional zone. The lack of spaces such as bedrooms results from the fact that for the research purposes, only the functional programs of the ground floors were taken into account.

The linear composition of the partial evaluation functions is a simplified model of multi-criteria optimization. In the near future, ELISi will be developed primarily based on the introduction of full multi-criteria optimization (using existing algorithms such as NSGA-II) and GUI full integration with the logical core of the proposed software.

Acknowledgements

The authors would like to thank Solene Veyseyre for her research materials regarding the America *favelas*. The authors would like to thank *Archipelag* design studio (Wrocław, Poland) for their contribution, support, and providing three architectural projects (*Flo*, *Simon*, and *Logan*) that increased the quality of our research.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Maciej Nisztuk  <https://orcid.org/0000-0001-6520-5128>

References

1. Mitchell WJ and Lo Dillon R. A polyomino assembly procedure for architectural floor planning. In: *Proceedings of the third Environmental Design Research Association conference*, Los Angeles, CA, 24–27 January 1972, Los Angeles, CA: University of California.
2. Mitchell WJ. A computer-aided approach to complex building layout problems. In: *EDRA2/1970 proceedings*, Pittsburgh, PA, 27–29 October 1970, p. 391. Saint Paul, MN: Environmental Design Research Association (EDRA).
3. Weinzapfel G and Negroponte N. Architecture-by-yourself: an experiment with computer graphics for house design. In: *ACM SIGGRAPH computer graphic proceedings*, Philadelphia, PA, 14–16 July 1976, pp. 74–78. New York: ACM.
4. Autodesk's The Living research studio, 2018, <http://www.autodeskresearch.com/groups/living/>

5. Nagy D, Villaggi L and Benjamin D. Beyond heuristics: a novel design space model for generative space planning in architecture. In: *ACADIA conference proceedings*, 2017, <https://autodeskresearch.com/publications/beyond-heuristics-novel-design-space-model-generative-space-planning-architecture>
6. Nagy D, Lau D, Locke J, et al. Project discover: an application of generative design for architectural space planning. In: *SIMAUD conference proceedings: symposium on simulation for architecture and urban design*, 2017, <https://www.autodeskresearch.com/publications/project-discover-application-generative-design-architectural-space-planning>
7. Nagy D, Villaggi L and Benjamin D. Generative urban design: integration of financial and energy design goals in a generative design workflow for residential neighborhood layout. In: *SIMAUD conference proceedings: symposium on simulation for architecture and urban design*, 2018, <https://www.autodeskresearch.com/publications/generative-urban-design-integration-financial-and-energy-design-goals-generative-design>
8. Merrell P, Schkufza E and Koltun V. Computer-generated residential building layouts. In: *ACM SIGGRAPH Asia conference proceedings*, Seoul, South Korea, 15–18 December 2010, article no. 181. New York: ACM.
9. Rodrigues E, Amaral AR, Gaspar AR, et al. GerAPlanO—a new building design tool: design generation, thermal assessment and performance optimization. In: *Energy for sustainability conference*, Coimbra, 14–15 May 2015.
10. Autodesk's AEC generative design. Project Fractal (Software), 2016. <https://home.fractal.live/>
11. Koenig R and Schneider S. Hierarchical structuring of layout problems in an interactive evolutionary layout system. *Artif Intell Eng Des Anal Manuf* 2012; 26: 129–142.
12. Nisztuk M and Myszkowski PB. Usability of contemporary tools for the computational design of architectural objects: review, features evaluation and reflection. *Int J Archit Comput* 2018; 16: 58–84.
13. Willis BR, Hemsath TL and Hardy S. A parametric multi-criterion housing typology. In: *32nd annual conference of Association for Computer Aided Design in Architecture (ACADIA)*, San Francisco, CA, 25–27 October 2012, pp. 501–510. San Francisco, CA: California College of the Arts.
14. Bao F, Yan D-M, Mitra NJ, et al. Generating and exploring good building layouts. *ACM T Graph* 2013; 32: 122.
15. Liu H, Yang Y-L, AlHalawani S, et al. Constraint-aware interior layout exploration for pre-cast concrete-based buildings. *Vis Comput* 2013; 29: 663–673.
16. Rodrigues E. *Automated floor plan design: generation, simulation, and optimization*. PhD Thesis, Department of Mechanical Engineering, University of Coimbra, Coimbra, 2014.
17. Knecht K and Koenig R. Generating floor plan layouts with K-d trees and evolutionary algorithms. In: *Generative art: proceedings of XIII generative art conference*, Milan, 16–17 December 2010, pp. 238–253. Milan: Domus Argenia Publisher.
18. Marson F and Musse SR. Automatic real-time generation of floor plans based on squarified treemaps algorithm. *Int J Comput Games Technol* 2010; 2010: 624817.
19. Ai T, Yu W and He Y. Generation of constrained network Voronoi diagram using linear tessellation and expansion method. *Comput Environ Urban Syst* 2015; 51: 83–96.
20. Koenig R, Miao Y, Knecht K, et al. Interactive urban synthesis: computational methods for fast prototyping of urban design proposals. In: Çağda G, Özkar M, Figen Gül L, et al. (eds) *Computer-aided architectural design: future trajectories*. Singapore: Springer, 2017, pp. 23–41.
21. Arvin SA and House DH. Making designs come alive: using physically based modeling techniques in space layout planning. In: Augenbroe G and Eastman C (eds) *Computers in building*. Boston, MA: Springer, 1999, pp. 245–262.
22. Tasadduq IA, Imam MH and Ahmad A. A hybrid algorithm for optimising facility layout. *South Afr J Ind Eng* 2015; 26(1): 120–134.
23. Dincer AE, Cagdas G and Tong H. A digital tool for customized mass housing design. In: *Proceedings of the 32nd international conference on education and research in computer aided architectural design*, Newcastle upon Tyne, 10–12 September 2014, pp. 201–211. Newcastle upon Tyne: Northumbria University.
24. Nisztuk M and Myszkowski PB. ELISI, evolutionary architectural aided design tool—analytical methodology of architectural design guidelines. In: Nisztuk M and Myszkowski PB (eds) *Shapes of logic: everything can be automated*. Wrocław: Oficyna Wydawnicza Politechniki Wrocławskiej, 2018, pp. 67–84.
25. Myszkowski PB, Olech LP, Laszczyk M, et al. Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. *Appl Soft Comput* 2018; 62: 1–14.

26. Archipelag Design Studio. Flo III G1, <https://www.archipelag.pl/?Page=Project&Project=flo-iii-g1> (accessed 21 May 2018).
27. Archipelag Design Studio. Logan G1, <https://www.archipelag.pl/?Page=Project&Project=logan-g1> (accessed 21 May 2018).
28. Archipelag Design Studio. Simon G2, <https://www.archipelag.pl/?Page=Project&Project=simon-g2> (accessed 21 May 2018).
29. Bianchini R. Le Corbusier—Villa Savoye part 1, history, <https://www.inexhibit.com/case-studies/le-corbusier-villa-savoye-part-1-history/> (2017, accessed 21 May 2018).
30. Veyseyre S. Case study: the unspoken rules of Favela construction. *ArchDaily*. 3 August 2014. <https://www.archdaily.com/531253/case-study-the-unspoken-rules-of-favela-construction> (2014, accessed 21 May 2018).
31. Neufert E and Neufert P. *Architects' data*. 4th ed. Chicester: Wiley-Blackwell, 2012.
32. Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T Evol Comput* 2002; 6: 182–197.
33. Myszkowski PB, Laszczyk M and Lichodij J. Efficient selection operators in NSGA-II for solving bi-objective multi-skill resource-constrained project scheduling problem. In: *Federated conference on computer science and information systems*, Prague, 3–6 September 2017, vol. 11, pp. 83–86. New York: IEEE.
34. Laszczyk M and Myszkowski PB. Improved selection in evolutionary multi-objective optimization of multi-skill resource-constrained project scheduling problem. *Inf Sci*. Epub ahead of print 2 January 2019. DOI: 10.1016/j.ins.2019.01.002.