Accepted Manuscript

Evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture Part 1: Methodology

Eugénio Rodrigues, Adélio Rodrigues Gaspar, Álvaro Gomes

PII: S0010-4485(13)00003-1
DOI: 10.1016/j.cad.2013.01.001
Reference: JCAD 2033

To appear in: *Computer-Aided Design*

Please cite this article as: Rodrigues E, Gaspar AR, Gomes Á. Evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture Part 1: Methodology. *Computer-Aided Design* (2013), doi:10.1016/j.cad.2013.01.001

Highlights

- A new hybrid evolutionary computation methodology and mathematical model are presented.
- The algorithm is tested in its validation and performance.
- The proposed technique generates floor plans to be used in the early architectural design stage.

Evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture Part 1: methodology

Eugénio Rodrigues[a,*], Adélio Rodrigues Gaspar[a], Álvaro Gomes[b]

[a]*ADAI–LAETA, Department of Mechanical Engineering, University of Coimbra*
Rua Luís Reis Santos, Pólo II, 3030-788 Coimbra, Portugal
[b]*INESCC, Department of Electrical and Computer Engineering, University of Coimbra*
Rua Luís Reis Santos, Pólo II, 3030-788 Coimbra, Portugal

Abstract

The drafting of floor plans is mostly hand made in today's architectural design process. The use of computerized floor planning techniques may enhance the practitioner's range of solutions and expedite the design process. However, despite the research work that has been carried out, the results obtained from these techniques do not convince many practitioners to accept them as part of their design methods. The existing literature shows that every research approach is different in the way in which architectural space planning is tackled. Consequently, each approach tends to be too specific or too abstract.

The Space Allocation Problem in architecture may be stated as the process of determining the position and size of several rooms and openings according to the user's specified design program requirements, and topological and geometric constraints in a two-dimensional space.

This is the first part of a paper that describes an enhanced hybrid evolutionary computation scheme that couples an Evolutionary Strategy (ES) with a Stochastic Hill Climbing (SHC) technique to generate a set of floor plans to be used in the early design stages of architectural practice. It presents the mathematical model with the problem statement and how the individuals' fitness is computed, the implemented methodological approach, how the adaptive operators are implemented, the summary of the overall procedure, and conclusions.

Keywords: evolutionary strategy, stochastic hill climbing, space allocation problem, space planning

1. Introduction

One of the first architectural design tasks is the drafting of floor plans which incorporate all the rooms in the design program according to the requirements and desires of the practitioner. The process is divided into two stages: analysis and synthesis. During the analysis stage, information and data about the design program are gathered, the equipment for each room is listed, functionality and requirements are determined and constraints are identified. During the synthesis stage, several sub-tasks are carried out which include setting up the design program in topological diagrams, sketching prototypical plans for each room, block planning, and drawing floor plans. This is a repetitive trial and error drawing process, where different elements are adjusted and rearranged, until a suitable design emerges respecting the requirements and constraints identified in the analysis stage. A large variety of floor plans may emerge at this stage and different potential solutions may be identified. The goal is, in an iterative process, to improve different designs and assess which solution is the most promising, according to the constraints, requirements and preferences of the practitioner.

This process may however become unbearable for humans if the complexity of the design program increases, making computers a useful and practical tool. Unlike humans, machines are capable of performing enormous amounts of repetitive routines without fatigue or error. What humans gain in creativity and innovation, they lose in productivity in repetitive tasks, also being prone to error. The purpose is not to replace creativity and innovation with productivity, but to employ a useful and user-friendly tool that is able to help the architect, by bypassing repetitive tasks, in the initial phase of the Space Allocation Problem.

Computational design synthesis has been applied to architecture since the 1960s. To some extent, these methods "emulated many of the habitual methods used by designers, with the added benefit of the computer's immense processing and storage capabilities" [1]. For Kalay, computational design synthesis methods may be classified into three main groups. The first two groups incorporate different contributions such as the use of simple generate-and-test algorithms, constraint-based approaches, assignment algorithms, grammar shapes, rectangle dissections with exhaustive enumeration, graph theory approaches, and knowledge-based systems. The use of these methods in engineering such as facility layout planning, VLSI

*Corresponding author.
Email address: eugenio.rodrigues@gmail.com (Eugénio Rodrigues)

layout planning, and bin packing, among others, produced good results, however, when faced with space planning, where the criteria are more subjective than objective, the results were disappointing. According to the same author, the techniques that fall within the last group, the Evolutionary Methods, mimic the evolutionary processes, which "have proved their ability to generate surprisingly novel solutions" and "the innovative abilities of GAs [Genetic Algorithms] have been demonstrated in part through their application to art and to the generation of floor plans" [1].

The Evolutionary Methods are capable of working with ill-defined and complex design problems [2]. Basically, they mimic the Charles Darwin theory of evolution by natural selection. Starting from an initial population of individuals (in space planning problem, individuals are floor plans that are potential solutions for the problem), evolutionary operators select, recombine and mutate the genetic material to produce the offspring. The process is repeated until one or more termination conditions are met. Potential solutions are assessed by calculating their performance in every evaluation axis (objective functions) and eventually, by taking into consideration other criteria such as the practitioners' preferences. In environments with multiple objectives, the individual's performance assessment is typically based on the Pareto non-dominance definition or on the aggregation of all objectives. The latter approach is carried out by using a weighted-sum method or by minimizing the distance to the goals that the decision maker wants to attain.

This paper presents an evolutionary algorithm approach for the Space Allocation Problem in architecture. The algorithm presents a hybrid behavior by combining an Evolutionary Strategy (ES) with a Stochastic Hill Climbing (SHC) technique. The purpose is to take advantage of both the global search capabilities of the ES and the local search characteristics of the SHC technique. This algorithm is named Evolutionary Program for Space Allocation Problem (EPSAP) and uses adaptive operators to perform the geometric transformations of the rooms, their walls and connections, and openings according to previously stored information.

The purpose of the technique is to help architects in the generation of a set of feasible floor plan designs by performing the repetitive tasks and generating a diverse set of alternatives which may be further improved and adjusted by the architects. This group of early design drafts is used to consolidate the preferences and choices of the practitioner, and they should be as diverse as possible but still respect the set objectives and constraints.

The EPSAP implements a two-stage search technique to achieve the goal of generating a diverse set of floor plan designs. This is accomplished by allowing each individual to evolve in his local region without sharing his genetic material. When the operators are incapable of further improvements, the individual is compared to the remaining population, and if it does not fall within the fittest, it is discarded and replaced by a new randomly generated floor plan to explore other regions of the search space.

The structure of this paper is as follows. The problem is presented in the "Introduction", the "Background" section classifies and analyses previous evolutionary approaches used to tackle the Space Allocation Problem in architecture and the "Mathematical model" section states the problem to be solved and how the individuals' fitness is computed. The following section, "Evolutionary approach: EPSAP algorithm" describes the algorithm and how the methods involved are coupled, and the "ES and SHC operators" section describes the geometric transformation operators that were employed. The technique is summarized in the "Overall procedure" section and, finally, the paper is concluded.

2. Background

Several evolutionary heuristic search methods have been used to tackle Space Allocation Problems in architecture (see Table 1 for the complete list of evolutionary methods used in architectural space planning with the respective objectives, design variables that were used, and topological features). These may be grouped into six types of approaches according to the model of the Space Allocation Problem and the purpose of each approach:

1. Area assignment: where the problem is modeled as a quadratic assignment problem in which a number of department unit areas must be assigned to an equal or larger building area. The methods used were Genetic Algorithms (GA) [3, 4, 10, 11, 13, 31, 32, 41] and Genetic Programming (GP) [12]. They usually work on a less detailed scale of space planning. Instead of dealing with spaces (rooms) the objective is to assign unit areas of each department (a set of spaces) to a building floor area. Consequently, these kinds of approaches discard individual spaces, circulation or openings, which are decided at a future stage of the design, or are already implicit as a constraint to be satisfied. For instance, it is possible to use a Dijkstra's Algorithm to determine the best position of the doors according to the shortest walking path [34, 35];
2. Area partitioning: a different approach in space planning problems is the partitioning of a building floor area in which the given area is to be divided into smaller areas. The design program must later be assigned to those areas. Several techniques may be used to divide the building floor area such as Voronoi Diagrams with a GA [36], K-dimensional trees with GA and ES [37–39], agent-based approaches with GP [28], and rectangle dissections on Non-dominated Sorted Multi-objective Genetic Algorithm (NSGA-II) [29, 42]. These approaches usually entail two phases. During the initial phase, the building area is subdivided into as many parts as the design program. During the second phase, the design program of the building is assigned according to topological requirements. Due to the fact that partitioning is made before the assignment of the design program, these approaches do not guarantee that the geometry will comply with the topological requirements.
3. Space allocation: used if the problem is to allocate spaces according to their topological relations and geometric constraints [20, 21, 24–26, 40]. A particular approach is the

Table 1: Comparison table of different approaches according to performance objectives, design variables, and topology features.

Approach		Perf.	Design variables									Top. feat.	
Authors	Methods	oF	wD	eD	eW	iD	S	fL	eF	bB	aB	oO	sA
[3, 4] Jo and Gero, 1996	GA	g					•	•		•			
[5, 6] Schnier and Gero, 1997	GA	g											
[7–9] Rosenman, 1997	GA	g					•						
[10, 11] Gero and Kazakov, 1997	GA	g					•	•		•			
[12] Jagielski and Gero, 1997	GP	g					•	•		•			
[13] Bentley, 1998	GA	g t						•		•			•
[14–16] Garza and Maher, 1999	GA	g t		•	•	•	•						•
[17–19] Elezkurtaj and Franck, 1999	GA/ES	g t				•	•						•
[20, 21] Michalek, 2001	GA+SA/SQP	g t h c l				•	•			•			•
[22] Jackson, 2002	GP+L-system	g											
[23] Virirakis, 2003	GP	g t		•	•	•	•		•				•
[24, 25] Makris, 2005	GA	g t					•						•
[26] Bausys and Pankrasovaite, 2005	GA	g h l				•	•			•			
[27] Homayouni, 2007	GA	g t					•						•
[28] Doulgerakis, 2007	GP	g t				•	•	•		•			•
[29] Banerjee et al., 2008	GA	g t					•			?			•
[30] Serag et al., 2008	GA	g											
[31, 32] Inoue and Takagi, 2008	GA+VD	g t l s					•			•			•
[33] Wong and Chan, 2009	GA	t											•
[34, 35] Thakur et al., 2010	GA/DA	g t s		•		•	•						•
[36] de la Barrera Poblete, 2010	GA+VD	g					•			•			
[37–39] Knecht, 2010	GA/ES+K-D tree	g t					•			•			•
[40] Flack, 2011	GA/GP	g t				•	•	•		•			•
Rodrigues, Gaspar and Gomes, 2012	ES+SHC	g t	•	•	•	•	•			•	•	•	•

GA genetic algorithms; GP genetic programming; ES evolutionary strategy; SA simulated annealing; SQP sequential quadratic progr.; L-System lindenmayer system; VD voronoi diagram; DA dijkstra's algorithm; SHC stochastic hill climbing; ? undetermined; • implemented; g geometric; t topological; h heating; c cooling; l lighting; s walk distance; oF objective function; wD walls dimensions; eD exterior doors; eW windows; iD interior doors; S space units; fL floor levels; eF equipment/furniture; bB building boundary; aB adjacent buildings; oO openings orientation; sA spaces adjacency

use of graphs, from graph theory, coupled with the GA to determine the best topological re-arrangement of the design program and to dimension the floor plan according to particular objectives. These objectives may be adjacency requirements and/or other design constraints as room ratios. One possible variation is to address only the topological constraints, for instance, the adjacency between rooms [33]. This will produce a set of feasible topological relations between spaces to be dimensioned. Following this step, other techniques such as linear programming may be used [43]. The inverse is also possible, for instance, using an ES to generate a set of solutions that comply with the geometric requirements, which evolve according to topological requirements using a GA [17–19].

4. Hierarchical construction: floor plans may also be understood as a hierarchy of elements. Starting from a discrete unit of space, one may join several units to give form to a room, several rooms form a zone, and several zones can be regarded as a floor plan [7–9, 27]. Different performance assessment criteria may be used to evaluate the relationship of the elements in different hierarchy levels. For instance, at the house level the topological adjacency requirements between rooms are assessed while at the room level what is evaluated is the ratio between area and perimeter [9]. This type of approach uses a discrete unit area from which everything is built. Meaning that the algorithm may produce designs with non-convex shapes. The non-convexity shape is related to the evaluation criteria set in the fitness function, for instance, penalizing according to the number of vertices in each room. Another approach is to comprehend the floor plan as a data structure of different objects, starting from the physical fixtures, equipment, and doors with their physical and functional area dimensions, to be assembled into room units, and these to be gathered into zone units, and finally combining the zone units into a building unit [23]. Each individual of the population is a variation of the building unit structure in their possible combinatorial re-arrangements.
5. Conceptual exploration: the method can be used to generate a set of solutions to explore designs in the conceptual exploration of ideas where the designer is no longer worried about requirements and constraints to be satisfied. Using a GP method coupled with a Lindenmayer System, the individuals of a population are assessed according to a "generic function" that represents certain subjective criteria of the designer [22]. It is also possible to use a GA with human evaluation to orientate the generation of biomorphic designs [30]. The objective is to inspire practitioners

by generating novel forms and configurations and not to draft the final floor plan.

6. Design adaptation: this approach consists in adapting previously stored designs to fulfill new requirements and constraints that have not been taken into consideration when the designs were produced [14–16]. The population of individuals are generated using the stored designs and are iteratively evolved until a satisfactory new design is obtained. This kind of approach attempts to speed up the computational search process by searching a similar space region, however, there is no guarantee that, according to the new requirements and constraints, the algorithm is not already in a local optimum or even that it will be possible to obtain a single solution. Basically, it is the search for the fittest individual in a very small search space region of an already drafted design. The algorithm initializes the population with a set of individuals that are small variations of a previously drafted floor plan design. The original design is already planned, as far as the topological relations and global positioning of its elements is concerned. Another variation of design adaptation is when the method is built to reproduce a particular style of architecture. Basically, it must have a two-phase approach. During the first phase, the search method develops a set of genes that are representative of specific architectural style elements, and these genes are used in the second phase to generate floor plans [5, 6]. In the last two approaches, the topological requirements are unnecessary to generate the floor plan as it is already specified in the original design or the purpose is to mimic an architectural style.

Architects have been reluctant to implement automated floor plan generation in their design process, despite the success of using optimization techniques in other fields of building design. The literature review highlights possible obstacles, as well as the natural rejection by practitioners to use any computational interference in the creative design process. The reviewed approaches seem to be too problem specific or too abstract. For example, some of them merely deal with topological aspects while others use the generation of forms to inspire (conceptual) the practitioner to further develop these.

The GA is the most common evolutionary algorithm used (see Table 1) sometimes hybridized with some other research techniques. However, preliminary tests with the algorithm proposed in this paper demonstrate that the use of basic crossover among different alternative solutions result in a less diverse set of floor plans, from the topological and geometric perspectives, and diversity is a key issue in early architectural tasks. In order to guarantee the integrity of the chromosome and the coherence of the floor plans, a judicious use of crossover must be carried out. For example, a common problem which arises when such computational tools are used is the duplication of spaces, or the lack of these, in comparison to the initial design program. This kind of problem is very often observed in different uses of GA in architectural floor plan design. Even if this problem is overcome by including a topology consistency check, after a few generations the individuals of the population converge to a similar solution and consequently, other design variations are lost. In the early design stages, when practitioners still have doubts about their choices, it is important that the generative tool does not limit their actions or point to a single solution, but enlarge the design scope.

Most of the reviewed works either tackled specific problems where the intention was to find the optimum solution, for instance, how to distribute departments within an already defined building boundary [4, 11, 12], or they were too abstract to explore the conceptual potential of automatic generation of forms [22, 30].

Some approaches only address the topological aspect of the design program [33] and leave the geometric implications of the spaces, openings, and walls dimensions for a later stage. The opposite also happens where topological concerns are secondary, for instance the orientation of openings, and in the case of topological relationships of space, some form of agent is used to identify which of the generated solutions are suitable [28, 37–39].

Table 1 lists the reviewed approaches. Each column identifies the objective function, design variables, and topological features and which method was used to tackle the situation. The objectives vary from work to work and these may be geometric, topological, related to energy efficiency (heating, cooling, and lighting) [20, 21, 26, 31, 32], or the walking distance between spaces [31, 32, 34, 35]. The design variables may be the wall dimensions, exterior doors, exterior windows, interior doors, space as a closed polygon, floor levels, furniture and fixtures, the building boundary, or adjacent buildings. The topological features are the orientation of openings, for instance the views from a window, and adjacency requirements between spaces. None of the listed approaches implements all of these.

All the approaches, except the algorithms, developed to generate new conceptual designs [22, 30] or to mimic architectural language [6], use some kind of orthogonal polygonal shape for each space that must be allocated, assigned, or partitioned. In addition to this, other architectural elements are added such as openings. Few of them deal with floor levels [4, 11–13, 28, 40].

One of the more complete works had a bottom-up approach to the problem of space planning [23]. It started by determining the shape of each space according to the functional requirements of necessary fixtures and furniture. After each space shape had been determined, they were assembled into a floor plan by combining the best position for every space. The main drawback of this approach is that as the number of rooms increased the space combinations also increased, the number of possible solutions is dependent on the initial space shape, and they could not work as modules that fit together. The researcher started from the unproven premise that if the functional area and relationship between objects are guaranteed then the assembled spaces will emerge as a floor plan. Each space may have several geometric forms. Determining the best shape for each space does not mean that the assembling of all spaces will result in a coherent floor plan. Despite the limitations of this kind of approach, it has the advantage of guaranteeing the allocation of furniture and fixtures from the start. The other approaches overcome this by setting minimum and maximum lengths for each

floor side shape.

Generally, in every research each space is treated as an orthogonal shape (rectangle) except in approaches inspired in biological morphology principles [22, 30]. Therefore, the use of non-convex shapes (for instance L-shaped spaces) is absent.

3. Mathematical model

The floor planning task in architecture is made up of several sub-tasks focusing on different aspects of the design process. The actual generation process in the synthesis stage depends on the goals and preferences of the practitioners' working methods. Therefore, there is no standard process for designing buildings for architects nor is there a standard process for floor planning.

When tackled by automated generation by computers, there are several approaches under the same name of space planning which have different purposes. As seen in the "Background" section, there are diverse approaches with different goals, from the simple conceptual exploration, where the generated floor plans hardly have any resemblance to an actual floor plan, to the highly specific optimization problems such as department assignment or adaptation of previously stored designs.

3.1. Problem statement

The Space Allocation Problem may be defined as a drawing of a set of predetermined rooms or spaces $I = \{S_1, S_2, \cdots, S_{N_s}\}$ on a two-dimensional space, which satisfy the required floor and openings dimensions, and topological relations within a building boundary $B = \{B_1, B_2, \cdots, B_{N_b}\}$ without overlapping these or adjacent buildings $A = \{A_1, A_2, \cdots, A_{N_a}\}$.

Each space $S_i(F_i, \{W_{i,1}, W_{i,2}, \cdots, W_{i,N_i^{ew}}\}, \{D_{i,1}, D_{i,2}, \cdots, D_{i,N_i^{ed}}\})$ is an object with one floor with the shape of a rectangle F_i, and N_i^{ew} exterior windows $W_{i,j}$ and N_i^{ed} exterior doors $D_{i,j}$. Each space can have topological preferences set by the user for the spaces (interior doors or adjacency) and openings (views). For instance, gathering a set of spaces in a cluster. This kind of preference allows the user to join common functional space units (e.g. bathrooms).

The floor is a rectangle with four degrees of freedom $F_i(x, y, w, h); x, y \in \mathbb{Z}; w, h \in \mathbb{N}$. x and y are the bottom-left vertex point coordinate, and w and h are width and height of the rectangle respectively. The width and height are constrained within the limits defined by the user, $a_n \leqslant w_i \leqslant a_m$ and $b_n \leqslant h_i \leqslant b_m$.

Each exterior window $W_{i,j}(s, p)$ and exterior door $D_{i,j}(s, p)$ has two degrees of freedom. The $s \in \mathbb{Z}_4$ specifies on which side of the floor every opening is placed and the $p \in \mathbb{Z}_{101}$ sets the relative position of the opening on that side. Each opening could have an orientation preference set by the user. This allows the user to specify preferences as far as views are concerned, which is a common practice for architects. This is guaranteed by allocating a vacant area in front of each exterior opening which must not be occupied by other elements.

The building boundary and adjacent buildings are sets of rectangles, $B_i(x, y, w, h)$ and $A_i(x, y, w, h)$ respectively, where $x, y \in \mathbb{Z}$ is the bottom-left vertex point coordinate, and $w, h \in \mathbb{N}$ are the width and height of the rectangle, respectively. In the case of the building boundary, the floor plan elements must be placed inside of the polygons resulting from the union of the B. Unlike the building boundary, the overlapping of the adjacent building set A must consider the vacant areas of the spaces and openings.

3.2. Computing individuals fitness

The purpose is to compute a set of feasible design solutions, N_{eg}, that minimize one cost function which is the sum of penalties due to the violation of constraints or by not fulfilling the topological and geometric objectives. The topological objectives are to satisfy the interior connectivity between spaces (interior doors), verify the adjacency of spaces and orientation of the exterior openings. The geometric objectives are to avoid overlapping between spaces, openings and adjacent buildings, set the correct dimension for the spaces, and to prevent the overflow of the building boundary.

The mentioned objectives are aggregated in a single objective function subject to minimization, expressed by Eq. 1, with seven evaluators named Connectivity/Adjacency (f_1), Spaces Overlap (f_2), Openings Overlap (f_3), Openings Orientation (f_4), Floor Dimensions (f_5), Compactness (f_6), and Overflow (f_7). All of the evaluators except the Connectivity/Adjacency Evaluator penalize areas, for instance the overlapped area between two spaces or the overflowed area of each space in relation to the building boundary. In the case of the first evaluator, the penalties are according to the distance between spaces. For example, the distance between two spaces that must be adjacent. Consequently, to have penalties with the same magnitude, the evaluators f_2 to f_7 are square rooted.

$$f(I) = c_1 f_1(I) + \sum_{i=2}^{7} c_i \sqrt{f_i(I)} \tag{1}$$

There are certain objectives which are essential when generating a floor plan. These objectives must be fulfilled entirely if a coherent floor plan solution is to be designed. The same is not so for the remaining objectives, as the success in observing this does not affect the actual floor plan. The first establishes the connectivity between spaces, avoids the overlapping of spaces and openings, and avoids the overflow of spaces relating to the building boundary. With this in mind, it is possible to set the importance of each objective in every evaluator's weight. For instance, the connectivity evaluator weight must be higher than the compactness evaluator weight. An adequate relationship among the weights of the evaluators is necessary to assure that the proposed technique is capable of producing good results and that the hierarchy of importance given by each weight to each evaluator is reflected. This must be satisfied at all costs.

Another aspect related to the evaluators' weights is that by attributing different values to two evaluators, for instance the higher value to the Connectivity/Adjacency Evaluator and the lower to the Compactness Evaluator, small improvements to the connectivity objective will compensate the penalties that

Table 2: Nomenclature.

Nomenclature			
P	Population of individuals I	N_p	Number of population individuals
I	Individual with a set of spaces S_i	N_s	Number of spaces in the floor plan
S_i	Space with a Floor F_i and a set of openings $W_{i,j}$ and $D_{i,j}$	F_i	Floor Rectangle of the space S_i
$W_{i,j}$ and $D_{i,j}$	Openings of the space S_i	N_i^{ew}	Number of exterior windows of a specific space i
N_i^{ed}	Number of exterior doors of a specific space i	E	Rectangle that bounds the floor plan
B	Set of building boundary rectangles B_i	A	Set of adjacent buildings rectangles A_i
N_b	Number of rectangles in the set B	N_a	Number of rectangles in the set A
$O(X_{i,j}, M(i))$	Clear area rectangle of the element X_i according to the clear area matrix M	c_i	Evaluator Weight
$V_x(X)$	List of all x-coordinates vertices of a rectangle X	$V_y(X)$	List of all y-coordinates vertices of a rectangle X
$w(X)$	Width of a rectangle X	$h(X)$	Height of a rectangle X
M_{con}	Connectivity/Adjacency Matrix	M_{dim}	Floor Dimensions Matrix
M_{ews}	Exterior Window Size Matrix	M_{eds}	Exterior Door Size Matrix
M_{ids}	Interior Door Size Matrix	M_{wa}	Exterior Window Vacant Area Matrix
M_{da}	Exterior Door Vacant Area Matrix	M_{far}	Floor Areas Matrix
M_{edo}	Exterior Window Orientation Matrix	M_{ewo}	Exterior Door Orientation Matrix
t_{iw}	Interior Wall Thickness	t_{ew}	Exterior Wall Thickness

could be obtained in the compactness objective when a geometric transformation operator is applied. This creates a kind of penalty buffer for the most important objectives.

The Connectivity/Adjacency Evaluator assesses the space distance according to the topological relationship set in a predetermined matrix M_{con}, the interior door dimensions set on the matrix M_{ids}, and the interior wall thickness on t_{iw} (see Fig. 1 for a topological graph of a floor plan design). If the value in the M_{con} matrix entry is 1 the connectivity is calculated, if it is 2 it is the adjacency which will be calculated, all this according to Eq. 2 which determines the penalties when requirements have been met. The f_{cdis} determines the connectivity distance between two spaces i and j, according to the necessary minimal distance of floor side to be juxtaposed. This will allow the placement of an interior door and it is expressed by Eq. 3. The distance of an x-coordinate (d_x) and y-coordinate (d_y) between two spaces i and j can be determined by Eq. 4 and Eq. 5 respectively.

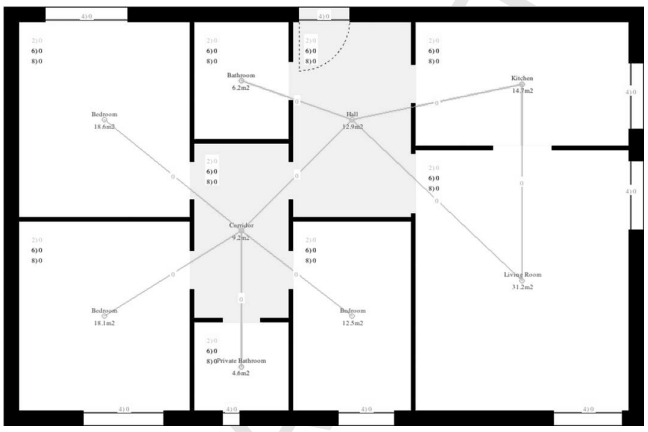

Figure 1: Example of a single house floor plan design with the penalties information and the spaces topological graph.

The Spaces Overlap Evaluator assesses overlapping among floor spaces or between each floor space (N_s) and adjacent buildings (set A). It is expressed by Eq. 6, where f_{ov} is the function to determine the overlapping area of two rectangles.

The Openings Overlap Evaluator assesses the overlapping of openings with other floor plan spaces and between the same space openings. If the vacant area in front of each opening is occupied by another element or adjacent buildings, the penalties will be the resulting intersection area. If one space has more than one opening, the performance of the individual will also be penalized if they overlap, corresponding to the overlapping area. Therefore, to determine the penalties for the overlapping of openings Eq. 8 is used. Where f_{sdw}, Eq. 9, determines the overlapping of spaces with the vacant area in front of openings, and f_{rdw}, Eq. 10, determines the overlapping of adjacent buildings with the vacant area in front of the openings. Where f_{dw}, Eq. 11, is used to compute the penalties for the overlapping of a space opening. To calculate the penalties of Eq. 9, Eq. 10 and Eq. 11 the values from the matrices for the vacant areas in front of the exterior windows and exterior doors are needed, M_{wa} and M_{da} respectively, where a rectangle dimension of the necessary area in front of the opening is established. The $O(X_{i,j}, M(i))$ expression returns the corresponding rectangle according to the $X_{i,j}$ opening and $M(i)$ entry value on the matching matrix. If the opening size is larger than what is established in the matrix the rectangle of the vacant area must be adjusted to cover the opening size dimension.

The Openings Orientation Evaluator penalizes each opening that is not placed on the preferred space floor side. If s, obtained from $W_{i,j}(s, p)$ or $D_{i,j}(s, p)$, is not equal to the entry on matrix $M_{ewo}(i, j)$ or $M_{edo}(i, j)$, respectively, the penalty value is the multiplication of the opening size times the depth value of the corresponding matrix vacant area entry. Eq. 12 calculates these penalties.

The Floor Dimensions Evaluator penalizes each floor space (F_i) that is over or under dimensioned relatively to the min-

$$f_1(I) = \begin{cases} \sum_{i=1}^{N_s-1} \sum_{j=1+i}^{N_s} f_{cdis}\left(F_i, F_j, t_{iw} + max\{M_{ids}(i), M_{ids}(j)\}\right) & \text{if } M_{con}(i,j) = 1 \\ \frac{1}{10} \sum_{i=1}^{N_s-1} \sum_{j=1+i}^{N_s} f_{cdis}\left(F_i, F_j, 0\right) & \text{if } M_{con}(i,j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$f_{cdis}(R_1, R_2, c) = \begin{cases} d_x(R_1,R_2) + d_y(R_1,R_2) + c & \text{if } d_x(R_1,R_2) \geq 0 \wedge d_y(R_1,R_2) \geq 0 \\ d_x(R_1,R_2) + d_y(R_1,R_2) + c & \text{if } d_x(R_1,R_2) \geq 0 \wedge d_y(R_1,R_2) + c \geq 0 \\ d_x(R_1,R_2) + d_y(R_1,R_2) + c & \text{if } d_x(R_1,R_2) + c \geq 0 \wedge d_y(R_1,R_2) \geq 0 \\ d_x(R_1,R_2) & \text{if } d_x(R_1,R_2) \geq 0 \wedge d_y(R_1,R_2) + c < 0 \\ d_y(R_1,R_2) & \text{if } d_x(R_1,R_2) + c < 0 \wedge d_y(R_1,R_2) \geq 0 \\ \min\left\{d_x(R_1,R_2) + c, d_y(R_1,R_2) + c\right\} - \max\left\{d_x(R_1,R_2), d_y(R_1,R_2)\right\} & \text{if } d_x(R_1,R_2) + c \geq 0 \wedge d_y(R_1,R_2) + c \geq 0 \\ |d_x(R_1,R_2)| & \text{if } d_x(R_1,R_2) + c \geq 0 \wedge d_y(R_1,R_2) + c < 0 \\ |d_y(R_1,R_2)| & \text{if } d_x(R_1,R_2) + c < 0 \wedge d_y(R_1,R_2) + c \geq 0 \\ \min\left\{|d_x(R_1,R_2)|, |d_y(R_1,R_2)|\right\} & \text{if } d_x(R_1,R_2) + c < 0 \wedge d_y(R_1,R_2) + c < 0 \end{cases} \tag{3}$$

$$d_x(R_1,R_2) = \max\left\{V_x(R_1), V_x(R_2)\right\} - \min\left\{V_x(R_1), V_x(R_2)\right\} - w(R_1) - w(R_2) \tag{4}$$

$$d_y(R_1,R_2) = \max\left\{V_y(R_1), V_y(R_2)\right\} - \min\left\{V_y(R_1), V_y(R_2)\right\} - h(R_1) - h(R_2) \tag{5}$$

$$f_2(I) = \sum_{i=1}^{N_s-1} \sum_{j=1+i}^{N_s} f_{ov}(F_i, F_j) + \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} f_{ov}(F_i, A_j) \tag{6}$$

$$f_{ov}(R_1,R_2) = w(R_1 \cap R_2) h(R_1 \cap R_2) \tag{7}$$

$$f_3(I) = f_{sdw}(I) + f_{rdw}(I) \tag{8}$$

$$f_{sdw}(I) = \sum_{i=1}^{N_s} \left(\sum_{j=1}^{N_i^{ed}} f_{ov}\left(F_i, O\left(D_{i,j}, M_{da}(i)\right)\right) + \sum_{j=1}^{N_i^{ew}} f_{ov}\left(F_i, O\left(W_{i,j}, M_{wa}(i)\right)\right) + f_{dw}(i) \right) \tag{9}$$

$$f_{rdw}(I) = \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} \left(\sum_{k=1}^{N_i^{ed}} f_{ov}\left(A_j, O\left(D_{i,j}, M_{da}(i)\right)\right) + \sum_{k=1}^{N_i^{ew}} f_{ov}\left(A_j, O\left(W_{i,j}, M_{wa}(i)\right)\right) \right) \tag{10}$$

$$\begin{aligned} f_{dw}(i) = & \sum_{j=1}^{N_i^{ew}-1} \sum_{k=1+j}^{N_i^{ew}} f_{ov}\left(O\left(W_{i,j}, M_{wa}(i,j)\right), O\left(W_{i,k}, M_{wa}(i,k)\right)\right) \\ & + \sum_{j=1}^{N_i^{ed}-1} \sum_{k=1+j}^{N_i^{ed}} f_{ov}\left(O\left(D_{i,j}, M_{da}(i,j)\right), O\left(D_{i,k}, M_{da}(i,k)\right)\right) \\ & + \sum_{j=1}^{N_i^{ed}} \sum_{k=1}^{N_i^{ew}} f_{ov}\left(O\left(D_{i,j}, M_{da}(i,j)\right), O\left(W_{i,k}, M_{wa}(i,k)\right)\right) \end{aligned} \tag{11}$$

$$f_4(I) = \sum_{i=1}^{N_s} \left(\sum_{j=1}^{N_i^{ew}} f_{or}\left(O\left(W_{i,j}, M_{wa}(i,j)\right), M_{ewo}(i,j)\right) + \sum_{j=1}^{N_i^{ed}} f_{or}\left(O\left(D_{i,j}, M_{da}(i,j)\right), M_{edo}(i,j)\right) \right) \tag{12}$$

$$f_{or}(O,z) = \begin{cases} w(O)h(O) & \text{if opening } s \neq z \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

imum and maximum dimensions for the smaller side (a_n and a_m) and larger side (b_n and b_m) of the floor rectangle. It also gives penalties if the referred space has an area inferior to the specified minimum area (f_{da}). Eq. 15 calculates the penalties of

this evaluator. The value of f_{sp} of a floor space is determined by Eq. 16. The Floor Dimensions Evaluator is computed by Eq. 14.

The Compactness Evaluator assesses the individuals by attributing penalty values for the empty area inside the building boundary (set of rectangles *B*). If it is an empty set, the penalty values are given for the clear area inside of an imaginary rectangle *E* that bounds the floor plan. Eq. 17 assesses how compact the individual is.

When the user establishes a polygon building boundary (the polygon is the sum of several rectangles, set of rectangles *B*) the Overflow Evaluator will assess if any space is partially or totally outside this polygon. However, before the assessment, the polygon building boundary is deflated according to the exterior wall thickness (t_{ew}) minus half the interior wall thickness (t_{iw}). This turns each floor rectangle into the core line of the walls. The equation to determine the overflow penalties is expressed by Eq. 18.

According to the problem statement, it is possible to determine the number of variables in a floor plan problem that are subject to change with Eq. 19. For example, a program design problem with 9 spaces, one exterior door, and each of the spaces with one window, the total number of variables to be computed within the geometric and topological constraints are 56.

However, the number of variables does not express the difficulties of the Space Allocation Problem in architecture. Certain difficulties arise from finding a way to quantify the subjectivity and preferences of the practitioner, which topological and geometric aspects should be taken into consideration, and the exponential computational resources required when the problem increases in its complexity. For example, the proposed technique expands the number of variables to include not only the positioning of the openings but to take into account the preferences of the user by pointing out the orientation of each one. Consequently, the computational burden increases and the proposed technique attempts to overcome this problem by cyclically renewing the population with new individuals to substitute those that have shown to be unfit and incapable of improving their fitness.

Regarding topological issues, the design program can be represented as a graph in which the rooms are vertices and interior doors are edges [33]. According to graph theory, it is possible to embed the graph within the plane if, and only if, no edges of that graph intersect. Due to the number of rooms and interior doors that connect them, stated by the user, the problem may become unfeasible if the topological requirements established by the user do not result in a planar graph, meaning that the produced design will have spaces overlapping or interior doors missing.

Again, depending on the user requirements, the geometric constraints may be such that it is impossible to have a feasible solution. For instance, if the interval of admissible space dimensions is so strict that no dimensional configuration is available to accommodate all spaces within the building boundary. Another example may be found when the same interval of admissible space dimensions does not consider the size of the respective openings, for instance the door size being larger than the side of the space. Consequently, the results will not include that opening or will resize it to fit.

The allocation and re-arrangement on the two-dimensional space of different objects is regarded as a combinatorial problem. The increase in the complexity of the problem is also the result of the exponential growth of the possible admissible solutions resulting from the expanded search space. This means that any algorithm that aims to carry out an exhaustive enumeration of the problem faces the limitation of the insufficient computation resources available.

4. Evolutionary approach: EPSAP algorithm

Due to the size of the search space and the combinatorial nature of the problem, computing suitable solutions may be a hard task. Evolutionary Methods have proven to be effective tools. However, as demonstrated in Table 1, very often such heuristic techniques only tackle a reduced set of design variables. Moreover, the results from some of these can barely be identified as floor plans, since different issues were not taken into consideration in the same approach. For instance, to ignore the role that exterior openings can have in the arrangement of the spaces.

With the purpose of enlarging the number of design variables, addressing different topological requirements, and still take the practioner's preferences into consideration, an ES based method was used. The proposed ES method seems to be adequate for the purpose of producing a set of design alternatives, leading to better results than with the GA and GP methods. The ES does not require the use of genetic operators which may result in incoherent floor plans. Doing so, mutation operators work directly on the phenotype level (physical representation). However, and according to the goal of the presented problem, a simple ES was not enough. It was necessary to guarantee that the individuals could be fairly compared before applying the elitism operator. In the initial population, individuals are randomly generated, merely producing loose rectangles distributed in the two-dimensional space. The number of transformations required until they form a coherent candidate solution varies from individual to individual. This requires the use of an SHC, which cyclically improves the individuals within each ES generation. The SHC transforms the individuals by applying geometric transformations to different design variables. Each operator will try to find the best position and size of an architectural element (for instance, the exterior window orientation).

The EPSAP algorithm is depicted in Fig. 2. The ES is an elitism approach that will filter the fittest individuals and replace the remaining by new randomly generated ones. The mutation operators of the ES will transform the individuals to emerge as floor plans by aligning their elements. The SHC on the other hand, transforms design variables according to geometric operations. This proposed technique joins the capability of the ES to search globally and the SHC to search locally.

The EPSAP is made up of two stages. The first stage is the ES search method in which the initial population of individuals is randomly generated with a population size set by Eq. 20.

The size of the population is related to the number of floor plan elements (exterior windows N_{ew}, exterior doors N_{ed}, and

$$f_5(I) = \sum_{i=1}^{N_s} \Big(f_{da}(i) + f_{sp}\Big(M_{dim}(i,a_n), M_{dim}(i,a_m), min\{w(F_i), h(F_i)\}\Big) \times max\{w(F_i), h(F_i)\} + f_{sp}\Big(M_{dim}(i,b_n), M_{dim}(i,b_m), max\{w(F_i), h(F_i)\} \times min\{w(F_i), h(F_i)\}\Big)\Big) \tag{14}$$

$$f_{da}(i) = \begin{cases} M_{far}(i) - w(F_i)h(F_i) & \text{if } M_{far}(i) > w(F_i)h(F_i) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$f_{sp}(n,m,d) = \begin{cases} (n-d)^3 & \text{if } n > d \\ d-m & \text{if } m < d \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$f_6(I) = \begin{cases} \sum_{i=1}^{N_b} w(B_i)h(B_i) - \sum_{i=1}^{N_b}\sum_{j=1}^{N_s}\Big(f_{ov}(B_i,F_j) - \sum_{k=1+j}^{N_s-1} f_{ov}(B_i, F_j \cap F_k)\Big) & \text{if } B \neq \emptyset \\ w(E)h(E) - \sum_{i=1}^{N_s}\Big(f_{ov}(E,F_i) - \sum_{j=1+i}^{N_s-1} f_{ov}(E, F_i \cap F_j)\Big) & \text{otherwise} \end{cases} \tag{17}$$

$$f_7(I) = \sum_{i=1}^{N_s} w(F_i)h(F_i) - \sum_{i=1}^{N_b}\sum_{j=1}^{N_s} f_{ov}(B_i, F_j) \tag{18}$$

$$f_{var}(I) = 4N_s + 2\sum_{i=1}^{N_s}(N_i^{ew} + N_i^{ed}) \tag{19}$$

interior doors N_{id}) multiplied by an adjustment factor k and the number of individuals on the elite group (N_{eg}). The equation does not consider the number of spaces (N_s), instead it uses the number of interior doors (N_{id}) as every space has one or more interior doors, making it more precise in representing the intricacy of the problem. This approach allows adjusting the population's size according to the complexity of the problem and to the needs of the user, allowing, at the same time, a diverse set of alternative solutions to be identified. According to the experiences carried out thus far, Eq. 20 permits an acceptable population size to be computed.

$$N_p = kN_{eg}(N_{ew} + N_{ed} + N_{id}) \tag{20}$$

The algorithm starts by initiating the first stage (ES search) by randomly generating the population. Then it initiates the second stage of the process (SHC search), in which individuals are evaluated and ranked according to the objective function expressed in Eq. 1. The assessment of the individuals focuses on seven evaluation axis: the connectivity/adjacency, floor overlapping, openings overlapping, openings orientation, floor dimensions, floor plan compactness, and spaces overflow. These are gathered in a weighted sum objective function to be minimized. The different weight permits each objective to be ranked according to its importance.

The following step of this stage is to determine if the termination condition is met for the second stage. If the moving average of the fitness of the elite group in the last 5 SHC search cycles is larger than the negative evaluators weights average obtained by Eq. 21, then the second stage finishes and returns to the first stage of the proposed technique (ES search). If not, the SHC search method continues by applying a group of SHC operators to each individual. Those operators perform stochastic geometric transformations in the individuals' openings, spaces clusters, and in the individual as a whole. Subsequently, the individuals are evaluated and ranked and the termination condition is evaluated again.

$$t = -\frac{\sum_{i=1}^{7} c_i}{7} \tag{21}$$

When the termination condition of the SHC search is met, the second stage finishes and the first stage is resumed. Continuing the ES search, the individuals are subject to a series of ES operators. This set of mutation operators are also stochastic geometric transformations, however, instead of dealing with each element of the floor plan at a time, work is performed on the individual by producing wall alignments or by removing empty areas around the perimeter. These ES operators are fundamental to the floor plan, to emerge as a unit and not as a sum of rectangles. They give the individuals the coherence of an architectural floor plan. Afterwards, they are again evaluated and ranked. The individuals which have a fitness value under the average fitness of the population are selected to be part of the next generation, making this an elitist approach. The remaining individuals are discarded and replaced by new randomly generated ones. Similar to the termination condition of the second stage, if the difference between the current generation of the

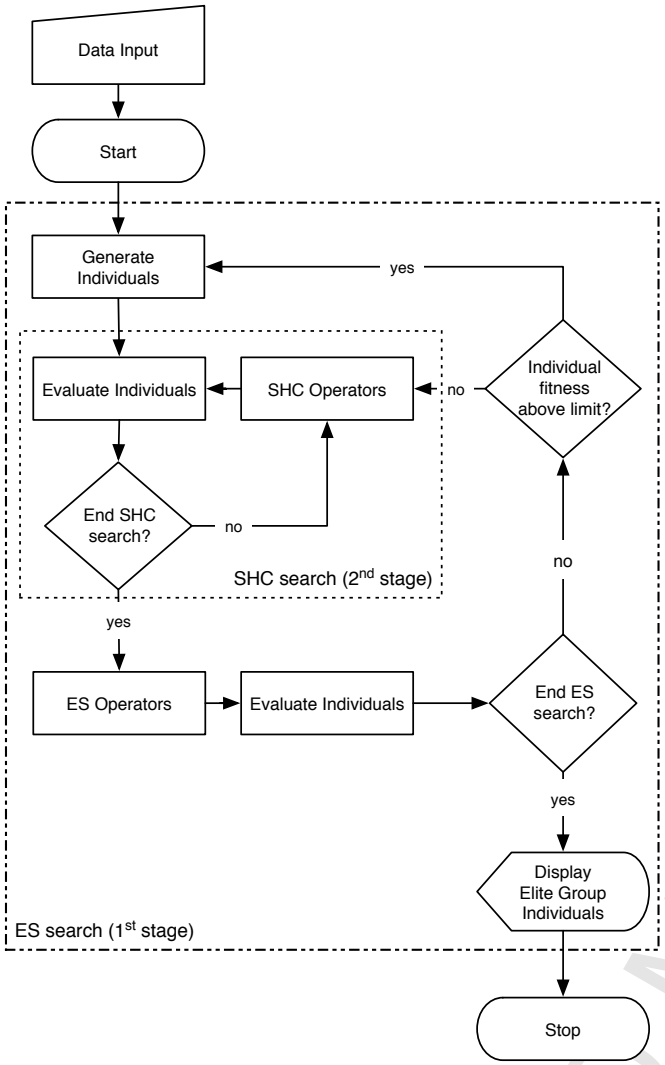

Figure 2: EPSAP flowchart.

elite group fitness average and the previous generation is larger than the negative evaluator weight average value calculated by Eq. 21, the search process ends and the individuals from the elite group are ready to be displayed.

The fact that the population is partially renewed with new individuals has two main advantages. It avoids the computational burden of continuous operations to evolve individuals that are no longer capable of improving and allows other search space regions to be explored, contributing to keep the evolution pressure on the elite group.

5. ES and SHC operators

As stated earlier, the proposed technique is made up of two search stages. During both stages, the individuals are subject to a set of adaptive stochastic operators that perform geometric transformations on the phenotype level of the individuals. After each operation has been carried out, the individual is evaluated and if an improved or equally fitted individual is produced, then the transformation is kept. The fitness information of each evaluator is stored and used in certain operators in adapting the magnitude of their transformation allowing for a faster convergence of the population.

The SHC operators arrange the openings and space position and dimensions such that topological and geometric constraints are satisfied, but these are not sufficient for the individual to emerge as an architectural floor plan. The ES operators are responsible for making the individuals as an architectural floor plan by removing incoherencies such as misalignment of walls, incongruent voids along its perimeter boundary or within the individual. The precise adjustment of these operators is essential for a fast convergence of the search technique.

5.1. SHC operators

The SHC operators are applied during the second stage (SHC search). Each individual will be subject to stochastic geometric transformations on their floors, openings, cluster of spaces, and on the floor plan.

Space operators modify the size and position of the floor spaces by translating, rotating, and stretching the floor of a random space of that individual floor plan. When the Floor Translation Operator is invoked, the floor is moved on the two-dimensional space in one of two possible directions, on the *x*- or the *y*-coordinate direction. The amount of translated distance is calculated by dividing the connectivity fitness by the connectivity/adjacency Evaluator weight and then multiplying by a random value from a Gaussian distribution. The Floor Rotation Operator uses the geometric center of the floor as a pivot for a turn of 90 degrees (counter-clockwise). The number of turns is randomly calculated within the set $\mathbb{N}_3$. The exterior openings attached to that space are also rotated according to the number of turns. The Floor Stretch Operator works by randomly picking a wall from the floor space and stretching it according to a random value from a Gaussian distribution.

Opening Operators reposition the exterior doors and exterior windows on the floor space. There are two operators of this kind. The first one sets a new random position on the floor space perimeter and the second mirrors the opening to the opposite side of the floor space. When the operator is invoked, one of the spaces is randomly chosen and one of the openings from that space is also randomly selected. If the Opening Translation Operator is applied, the algorithm will randomly choose the wall which will be placed and randomly determines the insertion point of that wall. The position of an opening, after the wall to be placed has been determined, is a random value (between 0 and 1) multiplied by the difference of the interior distance of adjacent walls with the opening size. The insertion point of the opening is the left edge in a clockwise orientation. In the case of the Opening Mirror Operator, the algorithm will place the opening in the opposite wall with the same distance from the same adjacent wall.

Two space cluster operators were implemented. The Spaces Cluster Translation Operator works by determining the higher connectivity penalties between spaces. Subsequently, the cluster of spaces associated to that connection is identified and

translated on an *x*-coordinate or *y*-coordinate direction by multiplying those penalties with a random value of the Gaussian distribution. Another situation where a cluster of spaces needs to be repositioned in relation to a specific space occurs when a cluster of spaces are stuck on the wrong side of that space. In this case, Spaces Cluster Mirror Operator is applied to allow the further evolution of the individual. This operator identifies the link between two spaces with higher connectivity penalties. After it determines which of the two spaces has the higher number of connections it mirrors this space and the connected spaces in relation to the center of the other space. The random direction can be on the *x*-coordinate or on *y*-coordinate and it is determined randomly. The center of the mirror is the connected space geometric center. The openings of those spaces are also mirrored preserving the relationship between the spaces and openings of that cluster.

The Floor Plan Operators work on the floor plan as a unit. There are four operators, however three are invoked only if the building boundary is set. They are the Floor Plan Centering Operator, Floor Plan Translation Operator and Floor Plan Rotation Operator. The first operator centers the floor plan within the building boundary. The second translates the entire floor randomly on the *x*-coordinate or on the *y*-coordinate. The magnitude of the translation is randomly determined within the interval 1 and the maximum value obtained by dividing the individual Overflow Evaluator penalties with the corresponding evaluator weight. This allows tiny adjustments in the positioning of the floor plan. The last operator rotates the entire floor plan relatively to the geometric center of the building boundary. The fourth is the Wall Translation Operator which acts on each iteration by randomly picking a wall, and randomly translating it in a perpendicular direction. The magnitude of this translation is obtained by randomly picking a value from the interval starting at 1 and the maximum admissible value from the division between the individual Floor Dimensions Evaluator penalties with the matching evaluator weight. Every time that a wall of a space is moved, the individual is assessed and if the individual fitness is equal or improved the transformation in that space is preserved.

5.2. ES operators

The ES operators are carried out in each ES generation. These operators act transversely to different floor plan elements, for instance the alignment of walls of different spaces.

These alignment operators act on the individual to remove incoherencies and make the floor plan emerge as a final design. The operators are invoked in the order: Wall Alignment Operator and Void Remover Operator. The former operator, the Wall Alignment Operator, works by making two lists, one with all vertices *x*-coordinates and the other with *y*-coordinates. If the ordinates are within the range of predefined values, they are substituted by their average. If the building boundary is set, the average value is substituted by the value of the boundary coordinates, if and only if, it is within a predetermined distance. In the latter case, when the Void Remover Operator is invoked, the voids of the plant are filled with the adjacent stretchable floor space, if the edge of the void is smaller than a pre-determined value.

6. Overall procedure

The proposed technique for finding a set of suitable floor plans for an early architectural design stage, according to geometric and topological requirements defined by the user, can be summarized as follows:

1. Start first stage of the technique (ES search). The initial population $P = \{I_1, I_2, \cdots, I_{N_p}\}$ is obtained by randomly generating individuals I.
2. Start second stage of the technique (SHC search).
3. For every individual in the population, the fitness is computed according to the objective function and its evaluators. See Eq. 1. The individuals are ranked and the elite group is created. Compute the elite group fitness average f_{Eavg} and store it.
4. If the difference from the current iteration f_{Eavg} to the average of the last 5 previously stored f_{Eavg} is greater than t, signal to stop the second stage and go to procedure number 6. See Eq. 21.
5. Apply SHC operators on each individual. Go to procedure number 3. If the individual has equal or improved fitness, the operation is preserved.
6. End second stage (SHC search) and return to first stage (ES search).
7. Apply the ES operators on each individual. If the individual has equal or improved fitness, the operation is preserved.
8. Evaluate individuals according to Eq. 1 and rank them. Compute and store the current elite group fitness average f_{CEavg} and the population average f_{avg}.
9. If the difference between f_{CEavg} and the previous generation f_{CEavg} is greater than t, signal to stop first stage of the technique and do procedure number 11. If not, start a new population generation with selected individuals that have their fitness lower than f_{avg}. Generate new random individuals to keep the same population size.
10. Restart second stage of the technique (SHC search). Go to procedure number 5 for the preserved individuals and go to procedure number 3 for the remaining.
11. Display the elite group individuals. Stop first stage of the technique.

7. Conclusions

The main purpose of the EPSAP is to be able to generate different candidate floor plans to be used in the early stages of an architectural design process. This will assist the architect in picking which possible solutions should be further developed at a later stage.

The EPSAP attempts to place both topological and geometric issues into a single search process. It is possible to gather several objectives in a single cost function, some of which are

mandatory and other are loose, and the latter may be adjusted according to the user's preferences. The geometric and topological features included, such as interior and exterior openings, their topological orientations, and walls thickness, have significant importance in how the spaces are adjusted to each other. It also guarantees that no opening will face, a wall or an adjacent building. These features are unique in the way that they were implemented in the EPSAP. The openings are participating objects of the evolving process and not just an after addition.

As it is oriented for the early design stage of architectural practice, it aims to generate a set of diverse solutions, thus becoming a helpful tool for the practitioner without narrowing their creativity and control over the tool. It uses the advantages of using an elitist ES technique to search a large space region and an SHC technique to locally improve each individual. The use of an elite group, corresponding to the user's desired number of floor plans to be compared, allows the algorithm to search further in search space. By discarding the unsuitable, the overpenalized individuals, and replacing them with new randomly generated ones allows for better use of computation resources.

The geometric transformation operators used in both stages of the process improve the individuals' fitness by randomly changing the design variables values. However, instead of proceeding with no determined function, the operators perform specific geometric manipulation. For example, the rotation of a random space in a random number of turns. Every time that an individual is evaluated, the detailed information is mapped into the individual itself and used to adapt the operators in the following transformation. The ES operators are important as they make the floor plans emerge as coherent solutions. They apply geometric transformations that are transverse to several floor plan elements, for instance wall alignments, removing incoherencies and voids in the design.

This part of the paper describes a hybrid evolutionary computation technique, in which an ES is enriched with a SHC method to tackle the Space Allocation Problem in architecture. It shows the structure of the algorithm, the different search methods involved, how the computation of the individuals' fitness is carried out and how the operators are executed in each individual of the population. In the second part of this paper, the proposed technique is tested for its validity and performance.

References

[1] Kalay, Y.E.. Architecture's New Media: Principles, Theories and Methods of Computer-Aided Design. Cambridge, Massachusetts: The MIT Press; 2004.

[2] Renner, G., Ekáart, A.. Genetic algorithms in computer aided design. Computer-Aided Design 2003;35(8):709–726. doi:10.1016/S0010-4485(03)00003-4.

[3] Jo, J.H., Gero, J.S.. Representation and use of design knowledge in evolutionary design. In: Tan, M., Teh, R., editors. Space layout planning using an evolutionary approach. National University of Singapore; 1996, p. 189–203.

[4] Jo, J.H., Gero, J.S.. Space layout planning using an evolutionary approach. Artificial Intelligence in Engineering 1998;12:149–162. doi:10.1016/S0954-1810(97)00037-X.

[5] Schnier, T., Gero, J.S.. Learning genetic representations as alternative to hand-coded shape grammars. In: Gero, J.S., Sudweeks, F., editors. Artificial Intelligence in Design '96. Kluwer; 1996, p. 39–57.

[6] Schnier, T., Gero, J.S.. Dominant and recessive genes in evolutionary systems applied to spatial reasoning. In: Sattar, A., editor. Proceedings of the 10th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence. 30 November - 4 December, London UK: Springer-Verlag London; 1997, p. 127–136.

[7] Rosenman, M.A.. The generation of form using an evolutionary approach. In: Evolutionary algorithms in engineering applications. Springer-Verlag Berlin Heidelberg; 1997, p. 69–85.

[8] Rosenman, M.A., Gero, J.S.. Evolving designs by generation useful complex gene structures. In: Bentley, P.J., editor. Evolutionary Design by Computers. Morgan Kaufmann Publishers Inc.; 1999, p. 345–364.

[9] Rosenman, M.A.. Evolutionary case-based design. In: Fonlupt, C., Hao, J.K., Lutton, E., Schoenauer, M., Ronald, E., editors. Artificial Evolution; vol. 1829 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-67846-5; 2000, p. 53–72.

[10] Gero, J.S., Kazakov, V.A.. Learning and re-using information in space layout planning problems using genetic engineering. Artificial Intelligence in Engineering 1997;11(3):329–334. doi:10.1016/S0954-1810(96)00051-9.

[11] Gero, J.S., Kazakov, V.A.. Evolving design genes in space layout planning problems. Artificial Intelligence in Engineering 1998;12(3):163–176. doi:10.1016/S0954-1810(97)00022-8.

[12] Jagielski, R., Gero, J.S.. A genetic programming approach to the space layout planning problem. CAAD Futures 1997;875:875–884.

[13] Bentley, P.J.. The revolution of evolution in design: from coffee tables to hospitals. In: Proceedings of Recent Advances in Soft Computing '98. 02-03 July, Leicester; 1998, p. 172–182.

[14] Garza, A.G.D.S., Maher, M.L.. Evolving design layout cases to satisfy feng shui constraints. In: Proceedings of The Fourth Conference on Computer Aided Architectural Design Research in Asia. 5-7 May, Shanghai, China; 1999, p. 115–124.

[15] Garza, A.G.D.S., Maher, M.L.. Gencad: a hybrid analogical/evolutionary model of creative design. In: Gero, J.S., Maher, M.L., editors. Computational and Cognitive Models of Creative Design V. Sydney, Australia: Key Center of Design Computing and Cognition, University of Sidney; 2001, p. 141–171.

[16] Garza, A.G.D.S., Maher, M.L.. An evolutionary approach to adapting design cases. Computación y Sistemas 2002;5:224–229.

[17] Elezkurtaj, T., Franck, G.. Genetic algorithms in support of creative architectural design. In: Architectural Computing from Turing to 2000. 1999, p. 645–651. 15-17 September 1999.

[18] Elezkurtaj, T., Franck, G.. Geometry and topology: a user-interface to artificial evolution in architectural design. In: 18th eCAADe Conference Proceedings. 2000, p. 309–312. 22-24 June 2000.

[19] Elezkurtaj, T., Franck, G.. Algorithmic support of creative architectural design. Umbau 19 2002;2:129–137.

[20] Michalek, J.J.. Interactive layout design optimization. Msc thesis; University of Michigan; 2001.

[21] Michalek, J.J., Papalambros, P.Y.. Interactive design optimization of architectural layouts. Engineering Optimization 2002;34:485–501. doi:10.1080/03052150214016.

[22] Jackson, H.. Toward a symbiotic coevolutionary approach to architecture. In: Bentley, P.J., Corne, D.W., editors. Creative Evolutionary Systems. Morgan Kaufmann; 2002, p. 299–313. doi:10.1016/B978-155860673-9/50049-5.

[23] Virirakis, L.. GENETICA: A computer language that supports general formal expression with evolving data structures. IEEE Transactions on Evolutionary Computation 2003;7(5):456–481. doi:10.1109/TEVC.2003.816581.

[24] Makris, D.. Study and realisation of a declarative system for modelling and generation of style with genetic algorithms: Application in architectural design. Phd thesis; Université de Limoges; 2005.

[25] Makris, D., Havoutis, I., Miaoulis, G., Plemenos, D.. MultiCAD - MOGA: A system for conceptual style design of buildings. In: Plemenos, D., editor. Proceedings of The 9th International Conference on Computer Graphics and Artificial Intelligence - 3ia2006. Limoges; 2006, p. 73–84.

[26] Bausys, R., Pankrasovaite, I.. Optimization of architectural layout by the improved genetic algorithm. Journal of Civil Engineering and Management 2005;XI:13–21.

[27] Homayouni, H.. A genetic algorithm approach to space layout planning optimization. Msc thesis; University of Washington; 2007.

[28] Doulgerakis, A.. Genetic programming + unfolding embryology in automated layout planning. Msc thesis; Bartlett School of Graduate Studies, University College London; 2007.
[29] Banerjee, A., Quiroz, J.C., Louis, S.J.. A model of creative design using collaborative interactive genetic algorithm. In: Gero, J.S., Goel, A.K., editors. Design Computing and Cognition 2008. Springer Science+Business Media B.V.; 2008, p. 397–416. doi:10.1007/978-1-4020-8728-8_21.
[30] Serag, A., Ono, S., Nakayama, S.. Using interactive evolutionary computation to generate creative building designs. Artificial Life and Robotics 2008;13(1):246–250. doi:10.1007/s10015-008-0588-3.
[31] Inoue, M., Takagi, H.. Layout Algorithm for an EC-based Room Layout Planning Support System. In: IEEE Conference on Soft Computing in Industrial Applications. Muroran, Japan; 2008, p. 165–170. doi:10.1109/SMCIA.2008.5045954.
[32] Inoue, M., Takagi, H.. EMO-based architectural room floor planning. In: 2009 IEEE International Conference on Systems, Man and Cybernetics. October; IEEE. ISBN 978-1-4244-2793-2; 2009, p. 518–523. doi:10.1109/ICSMC.2009.5346861.
[33] Wong, S.S.Y., Chan, K.C.C.. EvoArch: An evolutionary algorithm for architectural layout design. Computer-Aided Design 2009;41(9):649–667. doi:10.1016/j.cad.2009.04.005.
[34] Thakur, M.K., Kumari, M., Das, M.. Architectural layout planning using genetic algorithms. In: 3rd International Conference on Computer Science and Information Technology 2010, 9-11 July 2010; vol. 4. IEEE. ISBN 978-1-4244-5537-9; 2010, p. 5–11. doi:10.1109/ICCSIT.2010.5565165.
[35] Verma, M., Thakur, M.K.. Architectural space planning using genetic algorithms. In: The 2nd International Conference on Computer and Automation Engineering (ICCAE), 26-28 Feb. 2010; vol. 2. IEEE. ISBN 978-1-4244-5569-0; 2010, p. 268–275. doi:10.1109/ICCAE.2010.5451497.
[36] de la Barrera Poblete, C.I.. Algortimos genéticos como estrategia de diseño en arquitectura. Ph.D. thesis; Universitat Politécnica de Catalunya; 2010.
[37] Knecht, K.. Generating floor plan layouts with k-d trees and evolutionary algorithms. In: Soddu, C., editor. XIII Generative Art Conference. Politecnico di Milano University, Italy: Domus Argenia Publisher; 2010, p. 238–253.
[38] Knecht, K.. Generierung von grundriss-layouts mithilfe von evolutionären algorithmen und K-dimensionalen baumstrukturen. Informatik in der Architektur 2011;(9):35.
[39] Knecht, K., König, R.. Evolutionäre generierung von grundriss-layouts mithilfe von unterteilungsalgorithmen. Informatik in der Architektur 2011;(10):21.
[40] Flack, R.W.J.. Evolution of architectural floor plans. Tech. Rep.; Brock University; 2011.
[41] Bonnaire, X., Riff, M.C.. A self-adaptable distributed evolutionary algorithm to tackle space planning problems. In: Applied Parallel Computing; vol. 2367. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002, p. 403–410. doi:10.1007/3-540-48051-X_40.
[42] Quiroz, J.C., Louis, S.J., Banerjee, A., Dascalu, S.M.. Towards creative design using collaborative interactive genetic algorithms. In: Evolutionary Computation 2009. IEEE. ISBN 978-1-4244-2958-5; 2009, p. 1849–1856. doi:10.1109/CEC.2009.4983166.
[43] Damski, J., Gero, J.S.. An evolutionary approach to generating constraint-based space layout topologies. In: Junge, R., editor. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures. 4-6 August, Munchen, Germany: Kluwer Academic Publishers. ISBN 0-7923-4726-9; 1997, p. 855–874.

Short Curriculum Vitae

Eugénio Rodrigues is currently a PhD student for the Sustainable Energy Systems program (MIT-Portugal Program/Universidade de Coimbra). Received his degree in architecture from Technical University of Lisbon/Faculty of Architecture in 2002, and his advanced studies diploma in Urban Design from Lisbon University (ISCTE) in 2004. Works as full architect in Luís Coutinho Ramos Arqs. architectural office since 2008. His research interests are focused in automation, simulation and optimization procedures throughout the architectural design process.

Adélio R. Gaspar received his MSc and PhD in mechanical engineering from University of Coimbra (UC) in 1996 and 2004, respectively. Since 2004 he is Assistant Professor at the Mechanical Engineering Department, UC and full research of ADAI - Association for the Development of Industrial Aerodynamics. He is a certified experts and trainer of the Portuguese building energy certification system. Since 2010 he has been Member of the Executive commission for the revision of the Portuguese regulation for Energy Certification of commercial buildings (RSECE). His research interests include thermal simulation of buildings, HVAC, energy management of buildings and indoor thermal environments.

Álvaro Gomes received his Ph.D. degree from the University of Coimbra in 2004. He is an Auxiliary Professor at the Department of Electrical Engineering and Computers, University of Coimbra and a researcher at INESC Coimbra. His research interests include efficient use of energy resources, demand response, combinatorial optimization and evolutionary algorithms. He is an IEEE member since 1992.