Architectural Layout Evolution Through Similarity-Based Evaluation

N. Onur Sönmez

Architectural Layout Evolution Through Similarity-Based Evaluation

N. Onur Sönmez

We propose a novel application of similarity-based evaluation for layout design. Basing evaluations on a pool of existing designs or quickly delivered graphical specifications enables an Evolutionary Algorithm to become an open-ended system, where problem definitions can be kept open for fast updating. Through its flexible and open-ended aspects, the approach is able to develop draft layouts for an unlimited variety of tasks, which are not implied within the initial task definition. This enables a versatile computational layout design assistant, which can be easily adopted within regular design processes through usual graphic media. In this paper, the functioning of the approach is described and verified; the versatility of the resulting assistant, its intuitive graphical interface, and its usage scenario are demonstrated.

1. INTRODUCTION

Although they could ease the burden of the architects in the initiating of architectural plan layouts, artificial design assistants for such aims are still lacking in practice. After 40 years of effort, even the most ambitious studies depend on a few fixed, technical evaluation methods and strict and reductive problem definitions that prevent wider application. We see two basic problems with the existing approaches:

1. Architectural layout design has mostly been considered as an optimization problem, while it is not.
2. The developed evaluation approaches are not sensitive and dynamic as architecture requires.

In response, we propose a novel application of similarity-based evaluation for layout design, to be used within Evolutionary Computation. Through its flexible and open-ended aspects, the approach is able to develop draft layouts for an unlimited variety of tasks and enables a versatile layout design assistant, which can be easily adopted within regular design processes through usual graphic media.

In this paper, after a critical review of evolutionary layout design studies, the technical and philosophical bases of similarity-based evaluation will be discussed. In Section 3, the "design_proxy.layout" (d_p.layout) assistant will be introduced and its task definition and representation will be illustrated. Basic evolutionary aspects will be detailed in Section 4. The similarity-based evaluation approach will be demonstrated and verified in Section 5 and 6. In Section 7, the usage scenario of the d_p.layout will be illustrated. The strengths and weaknesses of the approach will be discussed in the concluding section.

2. LAYOUT PROBLEM AND A REVIEW OF RELATED STUDIES

Plan layout design (PLD) is a primary task in architecture. It concerns the topological and geometrical arrangement of activities within space, with an aim to meet a set of goals and criteria. At least since [1], architectural PLD has been frequently assumed as a variant of facilities planning or layout problems, which involves the placement of facilities in an industrial production setting. A related problem concerns 3D space allocation in warehouse and ship loading-unloading contexts. A high amount of research has been carried out in these areas with a variety of techniques, from algorithmic approaches to Simulated Annealing, Ant Colony Optimization, and Evolutionary Computation (EC). In this review, we will specifically focus on candidate evaluation approaches in EC, and within architectural contexts.

A series of early architectural layout systems followed constraint satisfaction approaches, such as HeGeL [2], WRIGHT [3], and SEED-Layout [4]. These systems evaluate candidate solutions under various constraints

concerning access, natural lighting, and privacy, as well as spatial, topological, functional, and geometric constraints. [5] presented an evolutionary system to produce space layout topologies. Fitness functions were based on connection, overlapping, and adjacency. [6] used an overall cost measure as the fitness function, which considers interaction between pairs of activities and the distance between their locations. Interaction matrix was hand-coded, based on subjective judgments of the client. Updating this study, [7] used cost of assigning an element, measure of interaction between elements, and measure of distance between elements as fitness criteria. [8] experimented with hierarchical interactive evolution. The criteria included perimeter-to-area ratio and the number of angles at the room level, and adjacency requirements between rooms at zone and house levels. In addition to quantitative assessments, qualitative assessments were carried out interactively by a user.

In more recent studies, [9] explored fitness functions that compare area, length-to-width ratio, connectivity of rooms, and perimeter-to-area ratios of houses. [10] separated the task into two phases; the first phase aimed at the generation of architectural layout topologies through EC, the second phase concerned the development of geometrical arrangements for given topologies through gradient based algorithms. Topology phase objectives comprised connectivity, Design Unit (DU) overlap, direct circulation path between DUs, and planarity. In the geometry generation phase, constraints functioned for prohibiting overlaps, providing access ways between spaces, forcing units to plan boundaries, and providing minimum space ratios, minimum construction material costs, feasible window sizes and minimum natural lighting. Additional objective functions measured heating cost, cooling cost, lighting cost, wasted space, and access ways, which were unified through a weighted sum approach. [11] represented adjacencies of DUs in graphs and evaluated the fitness of these graphs in terms of adjacency preferences, budget, and design constraints. [12] optimized given initial layouts using a pareto-based EA. Evaluations were based on given room dimensions and regularity of room shapes, circulation relations, legal window sizes, wall lengths (as a cost estimate) and plumbing lengths. [13] compared weighted, pareto-based, and rank-based approaches in multi-objective EC. Evaluation constraints included max-min numbers, sizes, areas, ratios, and types of rooms, having windows, proximity (connectivity) between rooms, and area comparisons between types of rooms. A criterion measured for the graph-depth average of the distance between entrance and each room. Additionally, a measure for closeness to "ideal geometric ratios" was included. Finally, [14-16] developed a detailed hybrid evolutionary technique, which couples Evolutionary Strategies (ES) with Stochastic Hill Climbing (SHC). Evaluation consisted of a weighted sum of all penalties for not satisfying the user requirements and/or constraints, which comprise, connectivity, adjacency, DU overlap, location on floor plan, clear area in front

of every opening (for doors and circulation), opening orientation, admissible dimension intervals, building boundary overflow, and compactness.

In most of these studies, the architectural PLD is structured as an optimization problem. User requirements and constraints comprise context-free numerical specifications for minimum areas, dimensions, adjacencies, connectivities, and openings. Due to practical problems, existing studies are mostly constrained to a small number of fitness criteria; while important contextual characteristics are mostly omitted (e.g., views, perceptions, etc.). An architectural layout is an integrated solution concerning a multitude of requirements on varying levels; functional, technological, perceptual, and intellectual. Because these are essentially complex and negotiatory, none of these issues can be expressed within fixed, context-free specifications. Even the technical aspects constitute highly complicated problems that result in extremely complicated solution processes. Furthermore, these too are context dependent. Consequently, real design processes progress through constant, dynamic, contextual re-evaluations [17]. This puts forward a requirement for more sensitive, flexible, contextual, and dynamic evaluation procedures. Holistic, state-recognizing (i.e., contextual), and similarity-based approaches can be a response for this predicament.

We propose that, most of the above-mentioned evaluation criteria could have been obtained from the formal characteristics of existing buildings. Any existing building exhibits a working collection of embedded choices and holistic solutions. Most of these are reflected through layout representations, which usually contain rich semantic references. We propose to base our evaluations on features of existing buildings through evaluation procedures that measure the similarity of proposals with target designs, where a series of layout features of existing buildings can serve as wholesale evaluation functions. [18-20] explored this possibility by using machine-learning techniques to generate implicit objective functions for usage in evolutionary generation of floor layouts. [21] evolved two-dimensional procedural textures using one or more target texture images that represent the desired texture features, such as color, shape and smoothness.

On a philosophical level, this should be seen as a preference towards the "family resemblance" conception [22], instead of objective functions based on numerical, explicit, and fixed specifications. Family resemblance and the related "thread" metaphor have been brought forward as an alternative to classes [22]. A class is an all or nothing concept, and is defined by a limited set of traits or definitions. However, many real world classification attempts raise insurmountable difficulties and the class concept is mostly unsatisfactory [23, p38]. The main problem is the difficulty of manually forming a limited list of required and sufficient traits that would precisely identify and differentiate a class of things. The traits that characterize things are numerous and hard to define. Worse still, exceptions appear everywhere. Compare these difficulties in determining ideal states for

buildings, as optimization approaches would demand. Consider also the difficulties that an attempt at a taxonomy of architectural typologies would encounter.

Family resemblance works through similarity or proximity to a paradigm case, or a prototype, which does not need to be ideal. From a cognitive science perspective, [24, p.77] claim that most concepts are not characterized by necessary and sufficient conditions, and prototype-based reasoning constitutes a large portion of actual human reasoning. In general, the structure of concepts includes prototypes of various sorts: typical cases, ideal cases, social stereotypes, salient exemplars, cognitive reference points, end points of graded scales, nightmare cases, and so on. Each type of prototype uses a distinct form of reasoning.

Family resemblance conception is robust and tolerant, it can work for typical and atypical examples alike, and it simply does not collapse. Just as family resemblance is an alternative to classification through a limited set of traits, similarity-based evaluation can be seen as an alternative to constraint-based evaluation approaches, rule-based state definitions, and approaches that measure a candidate's performance in terms of its relationship to predefined mathematical formulations and numerical specifications. Although constraints are useful for practically limiting a search space, it is indeed not possible to positively define a desired state only in terms of constraints. Moreover, it is practically impossible to define or evaluate design states with simple if-then style rules. Such approaches demand the definition of states through limited sets of explicit statements. For this to be possible, the defined states themselves have to exhibit a suitable ontological constitution. This is exactly what Wittgenstein was proposing in [25], and came to criticize in his later years with his family resemblance conception [22, p. 65-67]. This has been a recurring theme in Dreyfus' discussions; he puts forward the contrast between the two paradigms to support his case against AI, which he claims is won by him [23, 26, 27].

With the similarity-based evaluation approach, an objective function essentially remains incomplete. To tentatively complete its definition, it can use a pool of examples together with statistical techniques. Different objectives can use different example cases at the same evolutionary run. In addition, the evaluation information can be updated simply by adding or modifying example layouts without any updating or recoding on the system, hence information and procedures can be separated. The evaluation approach can also be improved by adding new analysis methods, which can be increased indefinitely. It should be noted that there would be a practical limit for the number of simultaneously followed objectives. The following sections will illustrate these claims.

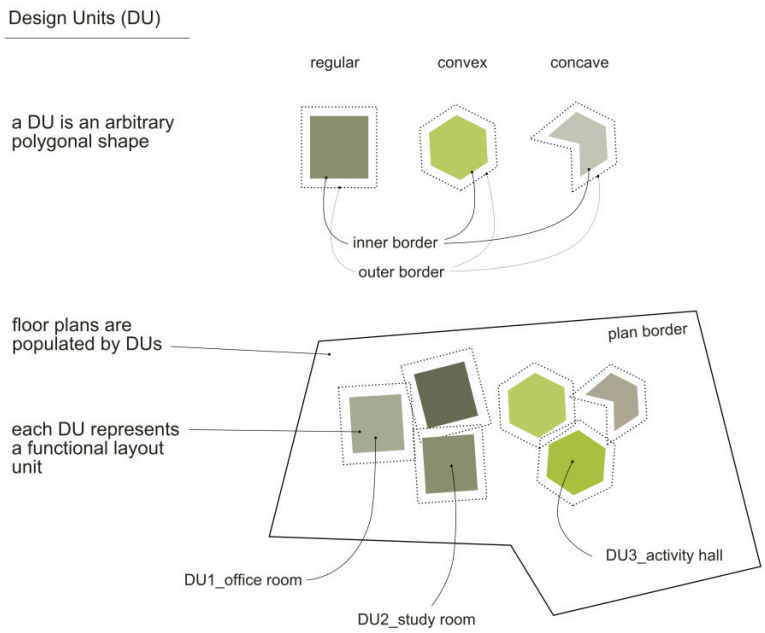
3. THE "DESIGN_PROXY.LAYOUT"

Similarity-based evaluation approach will be demonstrated through an architectural layout design assistant that we call "design_proxy.layout" (d_p.layout).[1] It participates in a regular architectural design process through its ability to generate draft layout arrangements that invite interpretation. A refinement functionality is excluded because this puts forward a much more complicated problem that requires a more sensitive contextual design intelligence, which is not available.

3.1. Task definition, representation, and interface

The task of the d_p.layout is to determine and arrange a list of arbitrary polygonal DUs over a series of building floors with variable plan boundaries within given contextual conditions and with regard to architectural considerations defined through given target layouts (Figure 1). The representation permits overlapping between DUs. Each DU has both an inner and an outer boundary. The inner boundary is used for measuring and penalizing overlap situations, while the outer boundary is used for detecting and rewarding neighborhoods and degree of proximity. Circulation elements are represented simply as another group of DUs. The layouts of a building are evolved consecutively. However, the DU list is composed and distributed all at once in the beginning of the whole process. During an evolutionary run, all candidate buildings share the same DU list with the same distribution to floors.

► Figure 1: Basics of problem representation.



[1]The application has been implemented with the Python language, using additional libraries, Scipy, Numpy, Polygon, Shapely, Matplotlib, Pycairo, and Rtree.

The d_p.layout carries out its task with regard to a series of example layouts (i.e., targets). A layering template is used for semantically marking (i.e., vector painting) example layouts to feed evaluation information to the system.[2] Through vector painting, parsing, and target preparation procedures, the example layouts are converted into targets for objective functions. User preferences and problem settings are defined through the same graphic interface, which registers basic characteristics of initial proposals (i.e., candidates). Through the same layering template with target vectorization, the user is able to define contextual information, floor borders, and optionally a series of fixed and mobile DUs. If there is remaining area in the candidate floors, the system will complete the DU list with reference to the DU distribution targets. The system delivers its draft layouts in the form of either bitmap or vector graphics that require and facilitate interpretation. The same color scheme is used consistently for all aspects of the process (Figure 14).

This constitutes a flexible interface, which, we claim, is intuitive, and is easy to adapt to a wide range of architectural scenarios with differing amounts and types of DUs, floor numbers, and plan outlines. As such, d_p.layout aims at offering a fast, flexible, and intuitive interface through usual design media. It supports varying degrees of user control; it can be used as a tool, as an assistant, or as something in between. This flexibility of use aims at pleasurable and playful usage.

3.2. Initiation and DU distribution

The DUs given in the initial DU list are distributed according to a probability map, which determines each DU type's probability of distribution to each of the floors. This map is prepared by using the DU type distributions of all chosen targets. Additionally, the system is able to fill the remaining areas on its own. For this option, an additional dictionary is prepared to bring together all the DUs of the target buildings with regard to their original floors. The aim is to utilize the ratios and dimensioning of the real DUs as given. The remaining DUs are drawn from this dictionary for each floor. Because of the differences in the scales of buildings, two parameters control the upper limits for the width and area/floor ratio of the DUs. When a large DU is drawn from the dictionary, the system iteratively divides this DU from its basic 'folds'. The value combinations of these folds are, (w/1, h/[1–3]), (w/2, h/[2–4]), (w/3, h/[3–5]), (w/4, h/[4–6]). When suitable smaller DUs are obtained, these are added to the floor until the area limit is reached or the floor is filled.

Fixed DUs are initiated at positions which are pre-specified by a designer. There is an additional option to initiate a series of movable DUs within specific locations to be modified during evolution. The remaining DUs

[2]Inkscape (an open source vector graphics application) is used for this interface.

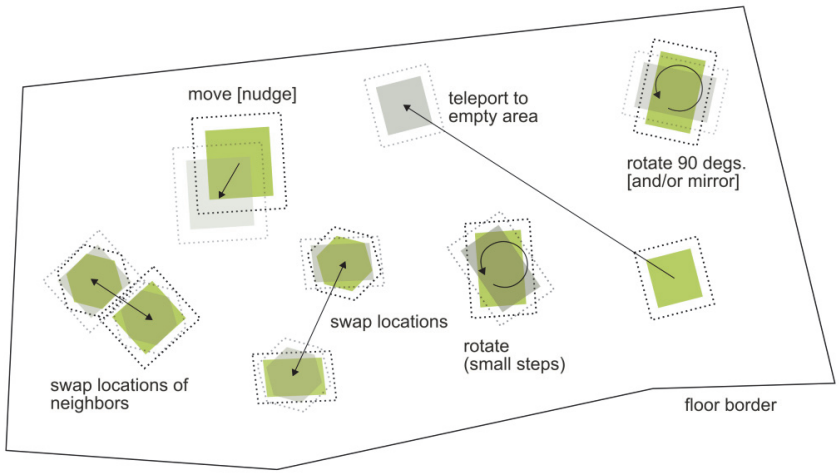
are initiated either within the boundary or within the bounding box of a given floor. Another option controls the initial rotation of the DUs according to the given floor boundary using the dominant orientation of the boundary lines.

4. BASIC EVOLUTIONARY ASPECTS

The d_p.layout uses a multi-objective evolutionary algorithm as its underlying generative mechanism, which we call "Interleaved EA" (IEA). As a variant of criterion-based multi-objective EAs [28], IEA enables the use of separate operators and settings for different objectives (see [29] for an early version). Each objective transforms the population until the fitness progression stagnates for this objective and then leaves the population to another objective's priorities. While the reproduction selections (for crossover and mutation) are carried out with respect to the dominating objective, the population selections use a rank-based operator, where all objectives are taken into account and the minimum rank number of an individual determines its selection probability. For each objective, each candidate has a separate rank amongst all candidates. The individuals are listed according to their minimum-ranks. Starting from the highest available minimum-rank, a desired number of individuals are selected. If a series of individuals occupy the same minimum-rank level, rank values of each individual are summed up, and the individuals are ordered in terms of the resulting values. Additionally, a simple approach is used to preserve diversity: if the fitness value of a new offspring already exists in the population, it is not inserted into the new population [30].

The genotype of a candidate building comprises separate floors. In each floor's genotype, there are fixed lists of fixed and movable DUs. For each DU, DU type, center coordinates, inner and outer polygon coordinates, bounding box, and related geometric information are stored. "N point crossover" is the default crossover operator. For reproduction selection, mostly uniform selection is preferred; however, tournament selection operator is also occasionally used.

The mutation operators (Figure 2) include Swap, Nudge, Neighbor swap (which only swaps DUs if they are legitimate neighbors, (i.e., outer boundaries overlap, while inner boundaries are not violated). There are two Rotate mutations; the first rotates a DU only for (±) 90 degrees. The second rotation mutation uses minor steps to rotate a DU incrementally, in order to adapt it to the orientation of neighboring borders and DUs. For Teleport mutation, empty spaces within plan boundaries are calculated and triangulated. Then a selected DU is translated to a random point within a randomly selected triangle.



◀ Figure 2: Mutation operators.

Each objective has a separate mutation roulette, which holds probabilities for each mutation operator. During the process, for each mutation candidate, a mutation operator is probabilistically drawn from this roulette. The operators, in turn, function stochastically through a set of parameters, which is also different for each objective.

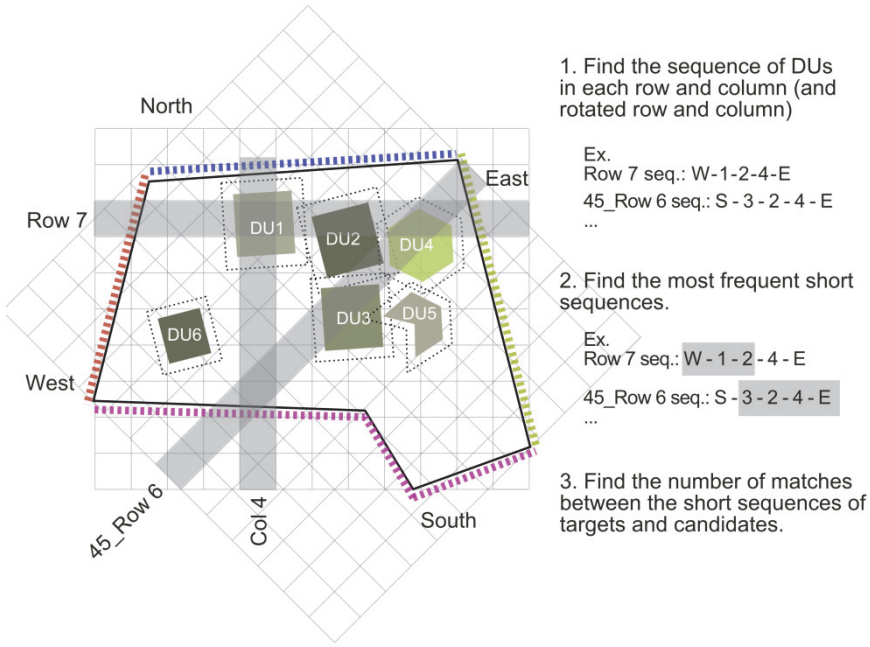
5. SIMILARITY-BASED EVALUATION

In total, six procedures will be presented as the objectives of the d_p.layout application. These are grouped as formal and functional objectives. Formal objectives keep DUs within borders while minimizing inner DU boundary overlap. "Trivial Hole" objective calculates the ratio of the occupied area within plan borders, so that maximizing this value amounts to the minimizing of DU overlap and outer boundary violation; and implicitly, minimizing the empty areas within boundaries. However, there is no explicit mechanism for eliminating trivial holes between DUs. "Out + overlap" objective gives penalty to floor boundary violation and to inner DU border overlaps, which are combined into a single fitness value by a weighted sum.

The functional objectives follow and exemplify the similarity-based evaluation approach, which is the focus of this paper. These will be described in detail, together with the verification of the functioning of each. Functional objectives are measured with reference to target buildings, which are prepared from the parsed information of example buildings before the evolutionary process.

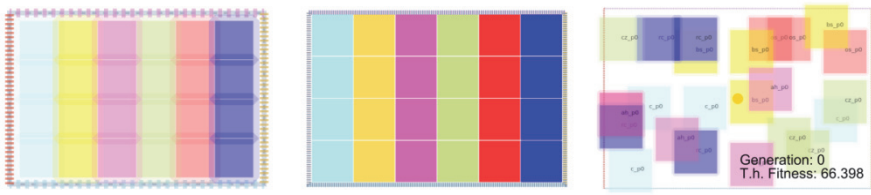
The "Sequence" objective (Figure 3) extracts sequences from target buildings and tries to develop new plans that predominantly include those sequences. Thus, the task is formulated as a sequence-matching procedure and is based on the methods of [31]. A Sequence target is prepared as follows: For each floor of an example building, rows and columns (i.e., boxes of a fixed width that are extended within the bounding box of a floor) are

► Figure 3: Calculation of Sequence fitness.



prepared. This operation is repeated for a 45° rotated coordinate system. For each of the rows and columns, all the DUs that collide with that row or column are found. Then the DUs are put in order from left to right. This yields a sorted sequence of DU types. If given, direction lines are also taken into account. Therefore, each sequence starts with a direction code and continues with DU types. Another direction code will be appended if a plan border is crossed again. This operation will continue until the end of the bounding box.

When all the rows and columns are processed this way, a series of 'long' sequences will be obtained. The width of the row or column determines the resolution of the analysis. Another parameter determines a set of lengths for short sequences (by default, sequences with length 3 to 5). All the long sequences are searched, for matching all the short sequences of this set of lengths. Finally, the most frequent (usually 100 to 200) short sequences and their reverses, together with found frequencies are taken as the sequence target. The length of the target influences the sensitivity of the target, but also lengthens the evaluation process. During evaluation, a candidate floor's long sequences are found in the same way, and each of the short sequences of the sequence target is sought within these. Whenever matching sub-sequences are found, a reward is given, which is proportionate with the frequency of that short target sequence. The fitness function is so designed that the sequences do not require an absolute matching of the target and candidate layout arrangements. It just tries to maximize the most frequent sequence types in the candidate, which makes the approach robust.

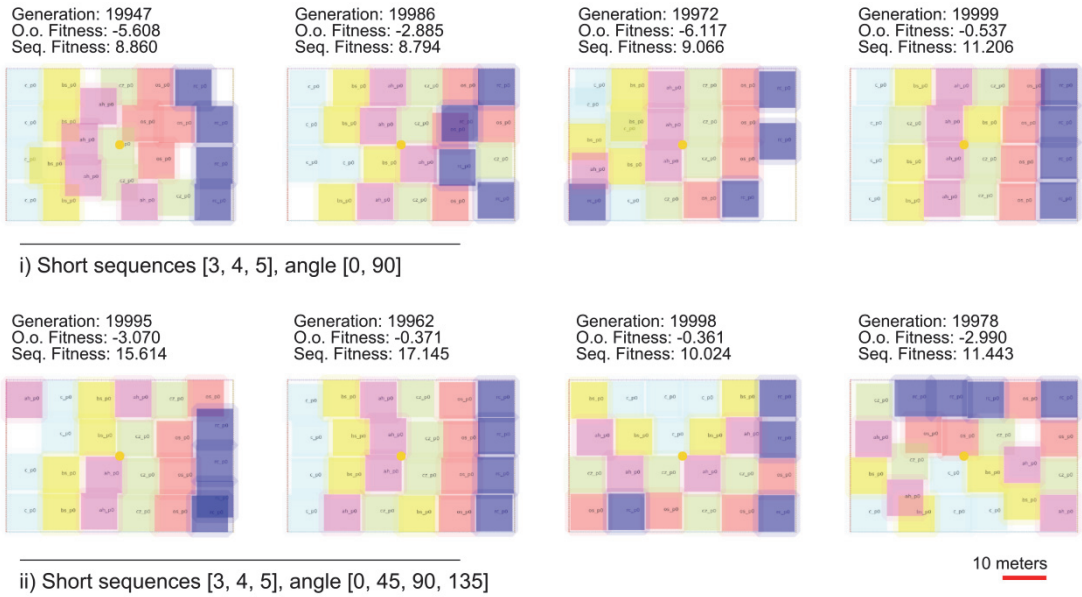


i) Sequence target ii) Candidate (Tight fit) iii) Initiation

◀ Figure 4: Target, candidate, and initiation for simple Sequence objective test.

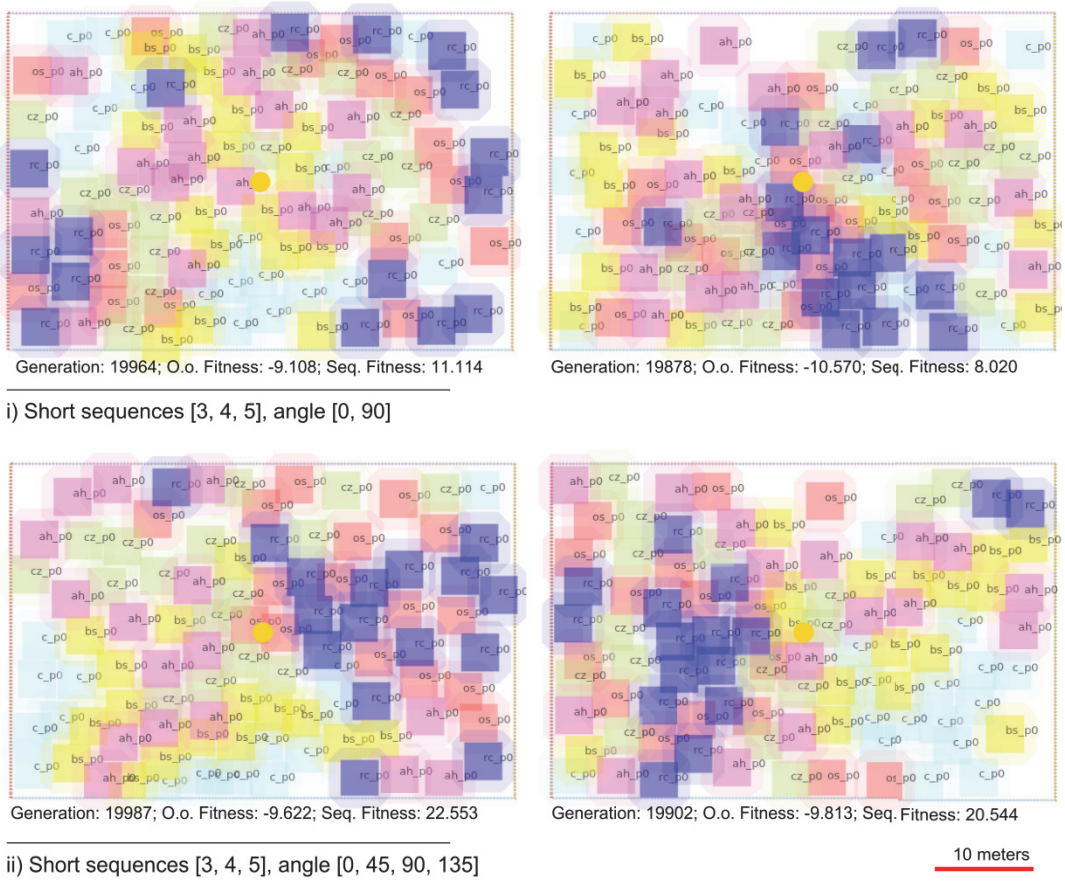
To verify the functioning of the approach, first, the simplest case will be illustrated, which comes close to searching for an absolute similarity of the target and candidate. Then through a more complicated case, the real functioning of the objective will be demonstrated. Target, candidate, and initiation for the first test series can be seen in Figure 4.

Best results of two test series are given in Figure 5 (population: 150, mutation: 15, crossover: 0; rotation mutations are disabled). Each image is the best Sequence objective result of a single evolutionary run. Visual examination reveals that the Sequence fitness is able to indicate the relative similarity of a target and a candidate. The visually fitter individuals can be seen to be also better in terms of their Sequence fitness values. The effect of the inclusion of direction lines is obvious; the obtained results mostly tend to follow the exact sequences, not the mirrored or rotated alternatives, which should have been obtained in equal numbers if the directions were not being considered.



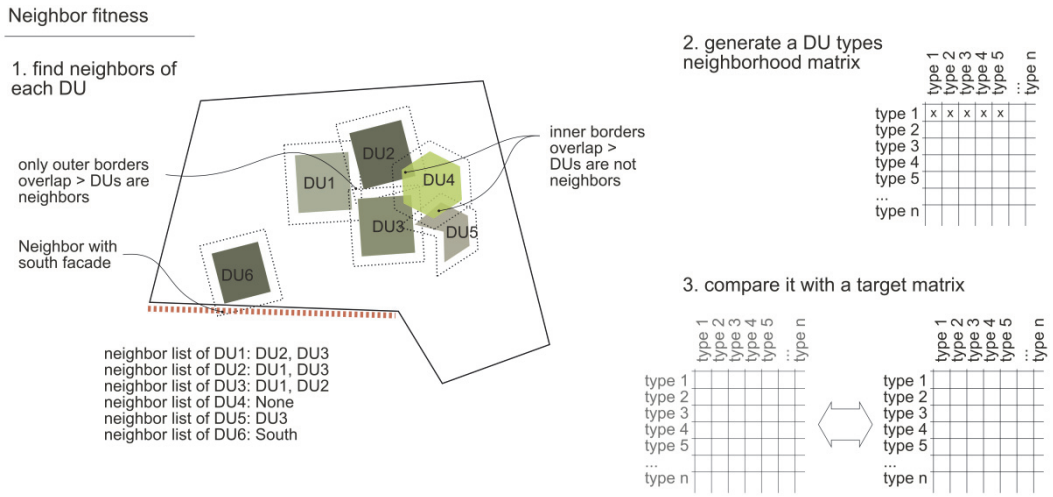
▲ Figure 5: Sequence objective verification – Sequence and Out + Overlap, best results of 0° and 0° + 45° versions – Sequence grid width = 2m.

The exact similarity pursued in the above trials is indeed not the desired behavior for the Sequence objective. Therefore, a second test series has been devised with 2x2 times the amount of DUs. Although more difficult in legibility, these series better assess the behavior of this objective. In Figure 6, on the upper row, the 0° tests are placed, while 0° + 45° tests are given below. In general, with more flexible problem setting, smaller fitness steps, and more resources to exploit, more regular and reliable evolutionary processes have been obtained. It can be observed that both the vertical same-color sequences and the horizontal color sequences were being generated in both 0° and 0° + 45° trials. Obviously, the 0° + 45° version has greater resources for exploitation, which results in conspicuous regular formations and higher fitness values. It can also be seen by visual inspection that the individuals that display more of the desired sequences obtain higher Sequence fitnesses.



▲ Figure 6: Sequence objective verification – Sequence and Out + Overlap – Sequence grid width= 2m.

The "Neighbor" objective tries to maximize the similarity of a candidate's neighborhood distribution to a target's (Figure 7). As such, it is thought as an alternative to adjacency matrices, which are usually static and are prepared by hand. This method enables the dynamic definition of a similar matrix through example buildings. A target is prepared as follows: The neighborhoods of example buildings are painted with a simple diagram. The direction that a DU is neighboring is also given with this diagram. Target preparation amounts to the collecting of the frequency of neighborhoods between DU types within a neighborhood matrix, from all floors of a single example building. The rows and columns of the matrix are symmetrical and their sequence is fixed for all applications. The frequencies of neighborhood types are normalized by the maximum frequency. This way the matrix becomes a neighborhood pattern, which can be compared with other buildings.

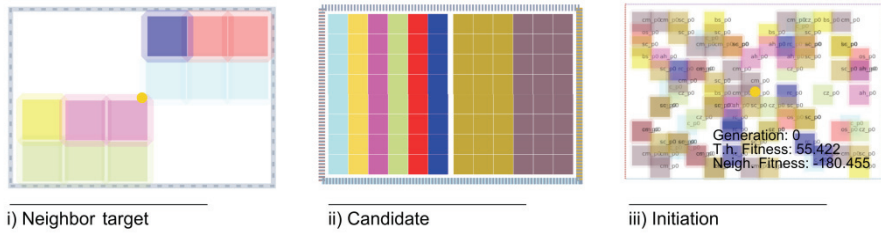


◀ Figure 7: Calculation of Neighbor fitness.

For candidate evaluation, neighborhoods of DUs are extracted from legitimate overlaps, i.e., overlap of outer boundaries while inner boundaries remain non-violated. Collision with the direction lines are used for the detection of direction neighborhoods. When the neighbors for each DU type are determined, this information is again converted into a normalized neighborhood pattern matrix. For fitness calculation, the absolute values of the differences of corresponding target and candidate matrix cells are summed up. The resulting value is negated to be able to treat this error value as positive fitness.

The goal of this objective is to bring and keep together several types of DUs with respect to target neighborhood patterns. Thus, for verification, the target consists of two almost distinct groupings (Figure 8), which are expected to reappear in the results. Half of the DUs in the candidate belong

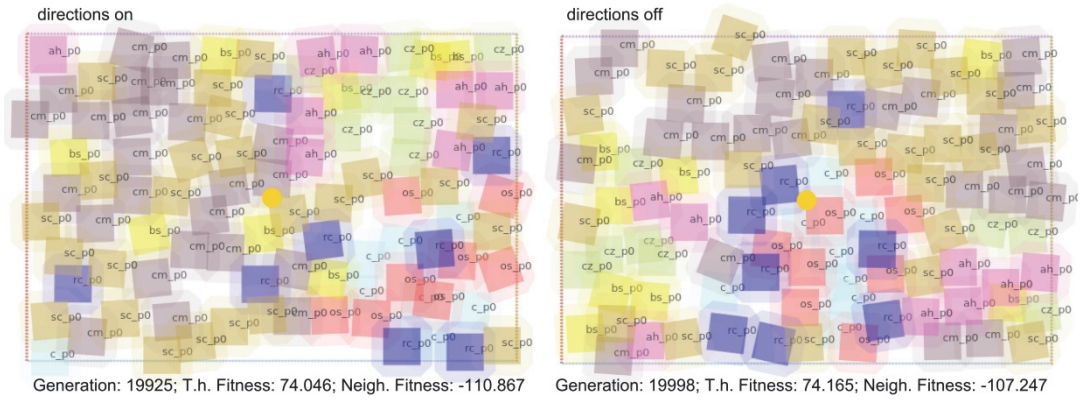
► Figure 8: Neighbor objective verification; target, candidate, and initiation.



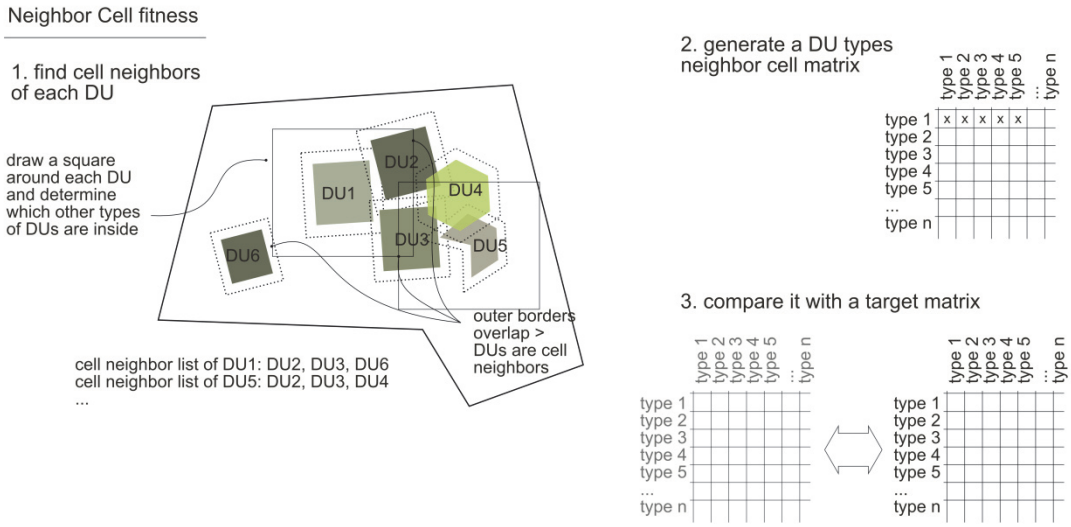
to the types in the target. The DU types of the other half have no target information. These are given with two colors (i.e., types) to see how the uninformed DUs will be distributed within resulting layouts.

Example best results are given in Figure 9 (population: 150, mutation: 15, crossover: 0). Left column gives the trials with directions enabled. The effect of directions appear negligible, which is understandable because the award for direction and DU neighborhood is the same and the system appears to exploit the more prolific DU-to-DU neighborhoods for this scenario.

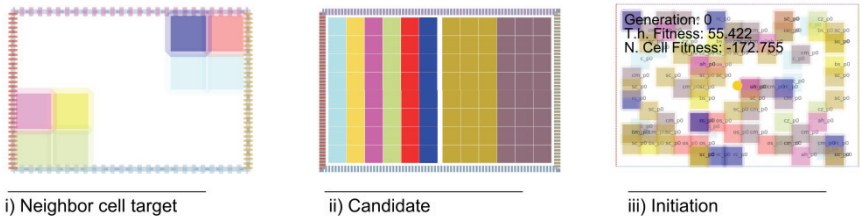
► Figure 9: Neighbor objective verification, example best results.



The difference of the "Neighbor Cell" objective from the Neighbor objective is the use of a fixed sized square query cell, placed over each DU's center (Figure 10). Types of all the DUs that collide with a DU's query cell are added to the cell neighbor list of that DU. For candidate evaluation, the target's matrix is compared with the candidate's. As with the Neighbor objective, Neighbor Cell objective calculates and negates the difference between two matrix patterns. Verification of the objective can be examined in Figures 11 and 12.

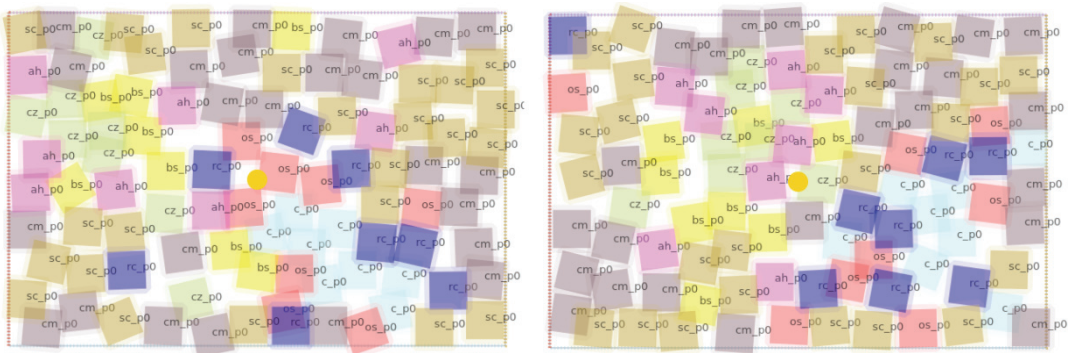


▲ Figure 10: Calculation of Neighbor Cell fitness.



◀ Figure 11: Neighbor Cell objective; target, candidate, and initiation.

▼ Figure 12: Neighbor Cell objective, example best results – Query cell width = 8m.



The last objective, "Manual Cell", tries to bring several DU types together by looking for cells that conform to manually given cell lists. This fitness is developed for obtaining already known functional groupings, like building cores, which frequently bring toilets, elevators, staircases, and technical elements together. The procedure regularly divides the bounding box of a candidate layout and traverses through each grid point with a square query cell for sampling the existing groupings. Each sample is compared with given desired cell groupings. Each query cell returns a ratio within range [0-1]. A ratio of "1" denotes that all desired DU types are found together within a cell, which ends the search procedure. Otherwise, the search continues through the whole grid and returns the best ratio that is found. If there are more than one desired cell lists, the average value of all the best-found ratios is returned. Manual Cell objective remains relatively simple compared to the other objectives, and desired fitness states are generally reached quickly.

6. MULTI-OBJECTIVE CASES FOR SIMILARITY-BASED EVALUATION

The multi-objective trials will aim at answering how the objectives function together for the evolution of layout drafts. This question will be addressed through a case that is derived from real-world examples. The candidate definitions follow the three floors of the Actur Library (Table 1). The vertical circulations and the semi open area on the ground floor are fixed.

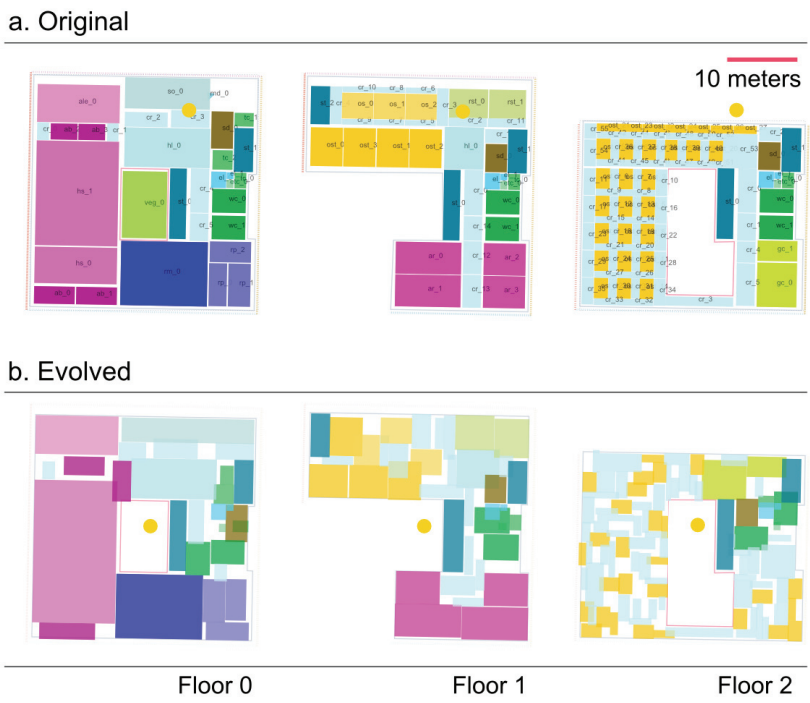
Table 1: Library buildings used as targets (All sources are accessed in July, 2014).

Name of Library	Architect	More Info
Actur Library, 2008	Carroquino Finner Arquitectos.	http://www.archdaily.com/23128/actur-library-carroquino-finner-arquitectos/
Des Moines Public Library, 2006	David Chipperfield Architects	http://www.davidchipperfield.co.uk/project/des_moines_public_library
Santa Monica Public Library, 2006	Moore Ruble Yudell Architects	http://www.moorerubleyudell.com/projects/santa-monica-public-library

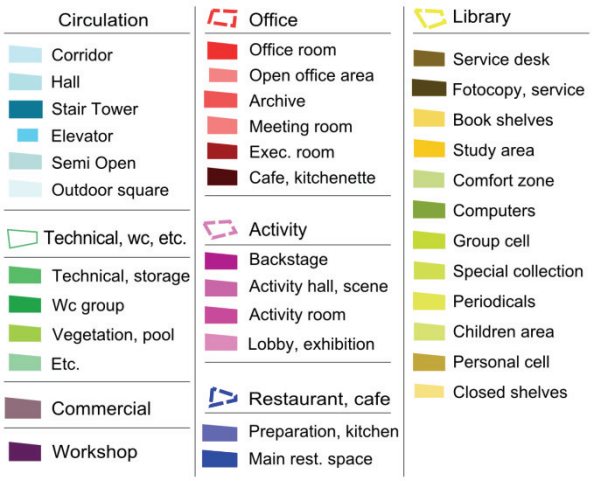
In all the following trials, an initial evolution is carried out with 6 objectives, which mostly develops the desired DU distributions. Population number starts from 5000, gradually diminishing to 150 in 20000 generations [30]. Crossover ratio is 0.02, and ¼ of the offspring are mutated. When the fitness improvement stagnates, a refinement run is carried out with 3 objectives (Trivial Hole, Out + Overlap, and Neighbor) over a smaller subset (150) of the latest generation, where crossover ratio is increased to 0.2, while ½ of these are mutated. For all trials, the Manuel Cell target is: ["stair

tower", "elevator", "technical"x3, "etc.", "wc"x2].

In the first test series, for the evolution of each floor, the corresponding floor of the Actur Library is used as the target (i.e., the ground floor's target is the ground floor of Actur Library). The candidate DUs are mostly identical to the targets. This generates a highly constrained problem definition, where the results are expected to approximate the originals. Example results can be seen in Figure 13 (the functions can be examined through the legend in Figure 14).

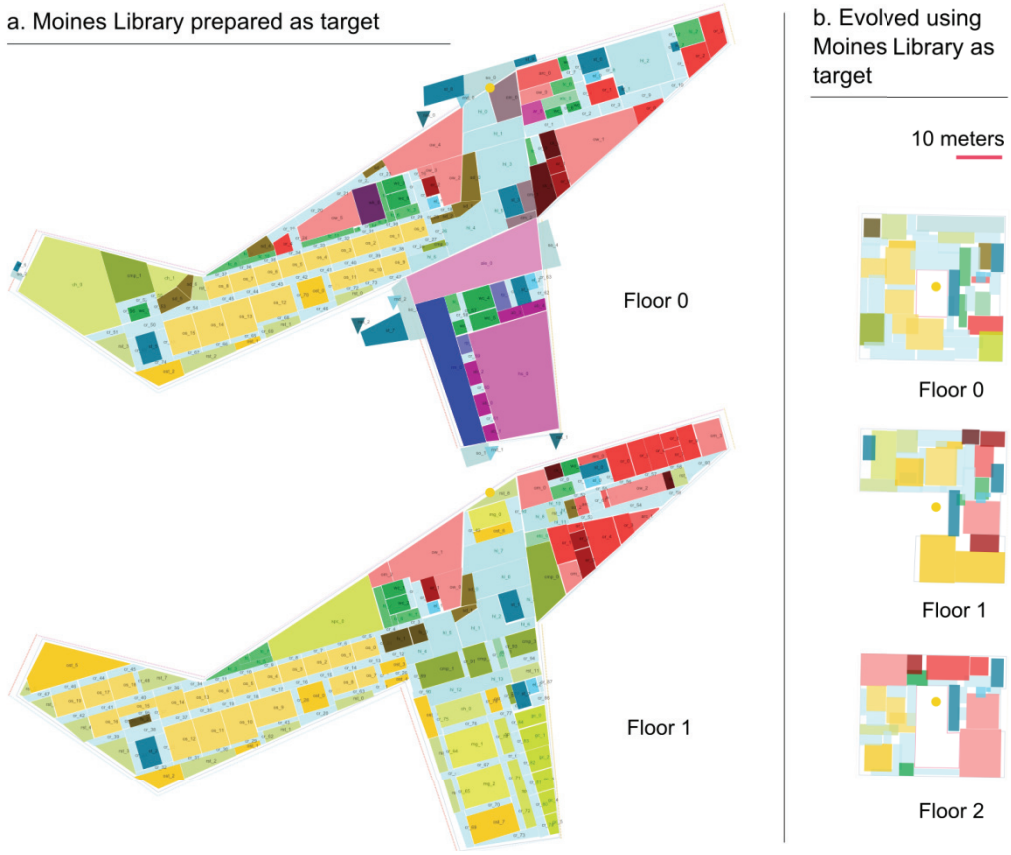


◀ Figure 13: From each floor of the Actur library, to each of the evolved floors.

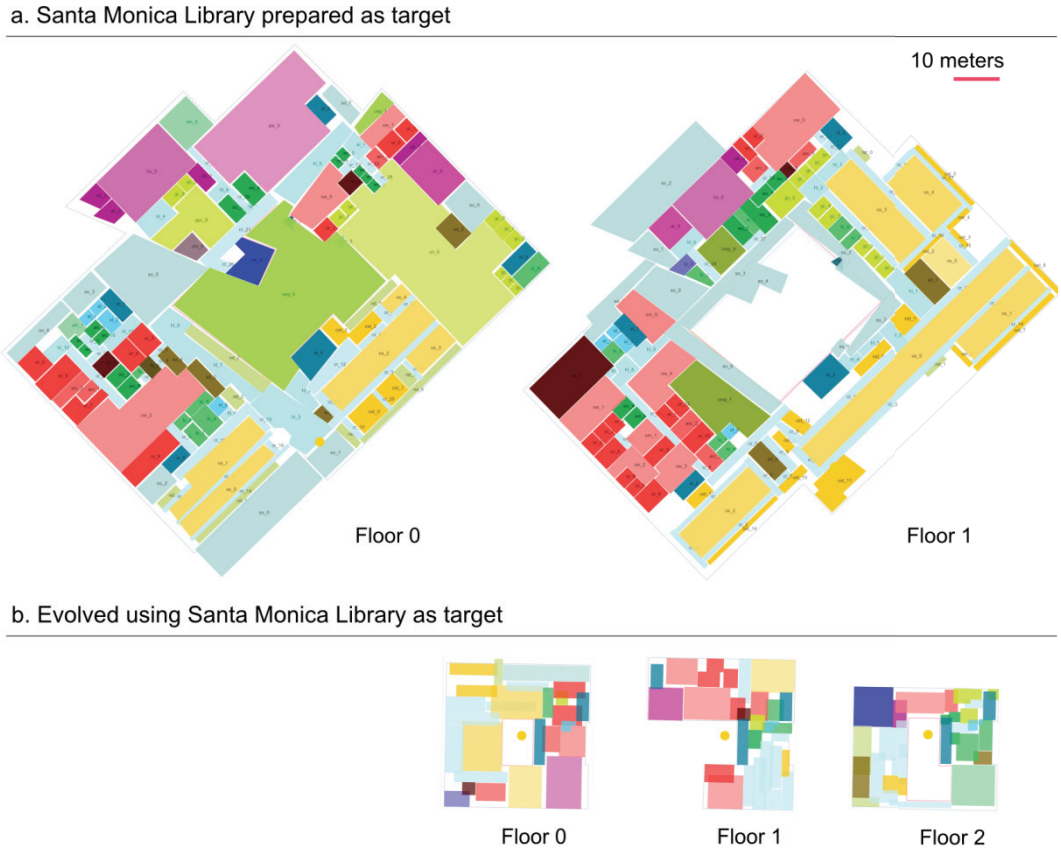


◀ Figure 14: Color legend for Design Unit (DU) types.

Obviously, this is not the real use case of the approach. The question is to develop new library plans by using the information of other libraries. Figure 15 and 16 present two cases, where the information on all the floors of an example library is used for generating the targets of the objectives. The DU lists are gathered from the target libraries in a probabilistic manner. Note the distribution and arrangement of the office, library, and service elements in comparison with the targets (Figure 15, 16). As can be seen, the system arrives at proposals in any case. Although the drafts lack overall circulation organization, they nevertheless develop a reasonable placement and grouping of the DU types. It is advisable to fix main halls and main circulation elements together with the vertical elements.



► Figure 15: From Moines Library to floor outlines of Actur Library.



▲ Figure 16: Santa Monica Library to Actur Library.

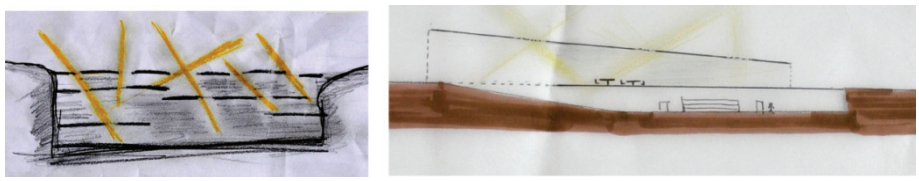
7. USAGE CASE OF THE D_P.LAYOUT

Two student workshops have been carried out with students from varying levels (from first year to graduate), to test if the d_p.layout is practically useful, easy to use, and pleasurable. Both workshops have been shaped and presented as regular architectural design workshops, i.e., not as computational design workshops. The focus was on the design of library buildings at specific sites. The core design steps of the workshops comprised fast collective brainstorming sessions; study, presentation, and discussion of precedents; exercises that spatially enrich resulting buildings; and a fast progression towards a detailing of 1/200 models. In both workshops, after a short tutorial session of around an hour, and with some practice, the students quickly became able to define their outlines and contextual issues.

As an example case, the design process of Merve Bahur, Bahri Özgötürücü, and Melike Kaya will be illustrated. This team's initial keyword was "depth". Setting out from this word, they decided to emphasize the depth of a building by a gradual passage from brightly illuminated spaces to

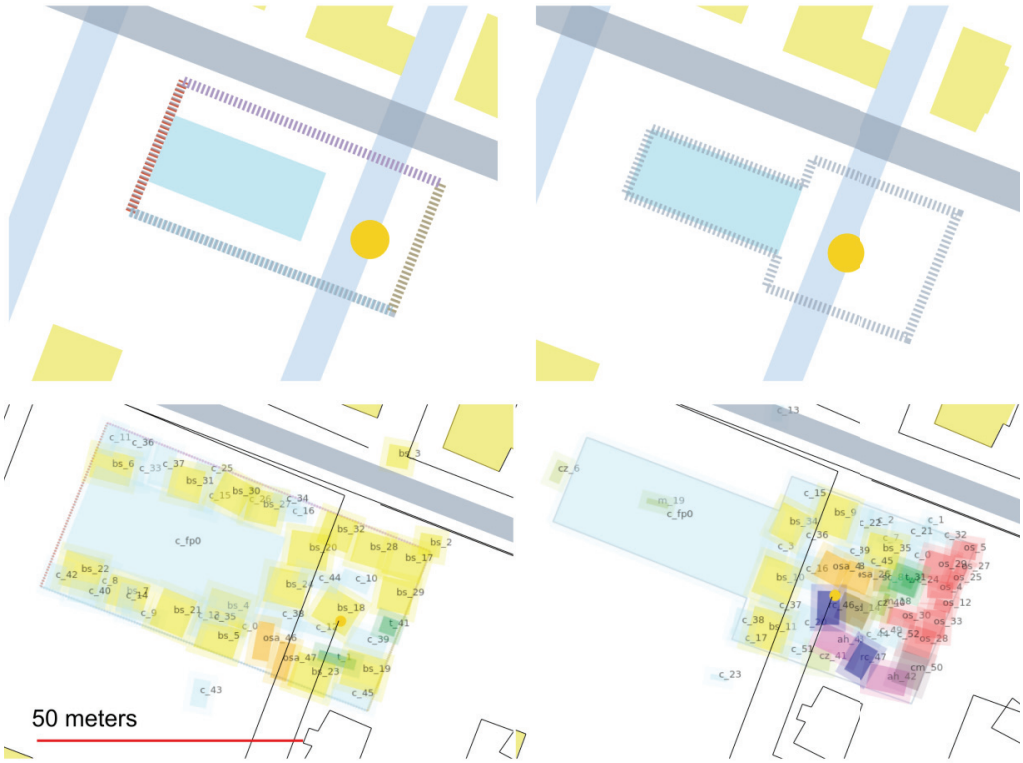
the darker depths of a building. This was to be realized by a ceremonial procession to the underground through a dominant ramp (Figure 17). The outer envelope would be perforated by regular rifts, which were to continue with lesser density on the ground floor slab, so that there would be two light-filters, which would separate spaces in terms of lighting conditions.

► Figure 17: Merve, Bahri, and Melike, conceptual sections.



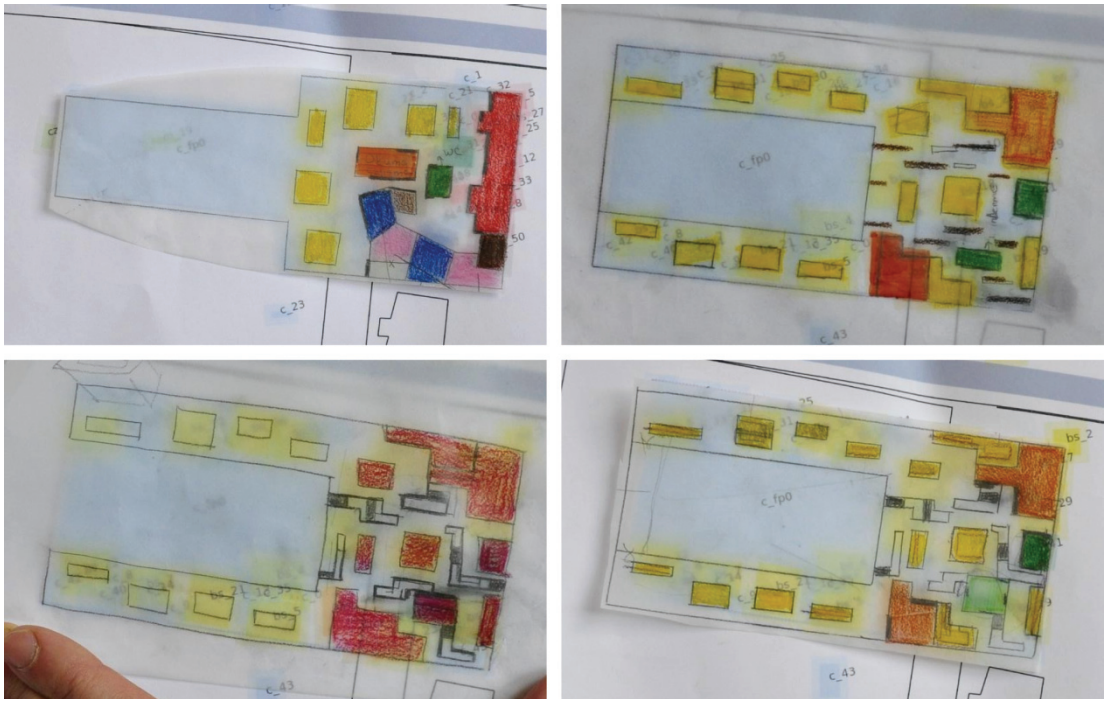
They prepared their d_p.layout proposal by fixing a hall that would represent the large entrance ramp on both ground and basement floors (Figure 18). Resulting d_p.layout drafts appeared mostly applicable and they became instructive for the team while developing their final proposal, which resulted in dynamic and convenient interiors. The team decided to keep the bookshelves on the side aisles within floating bookshelf cubicles.

► Figure 18: Merve, Bahri, and Melike, d_p.layout preparation and results (DU area limit = 100m^2, DU width limit = 8m).

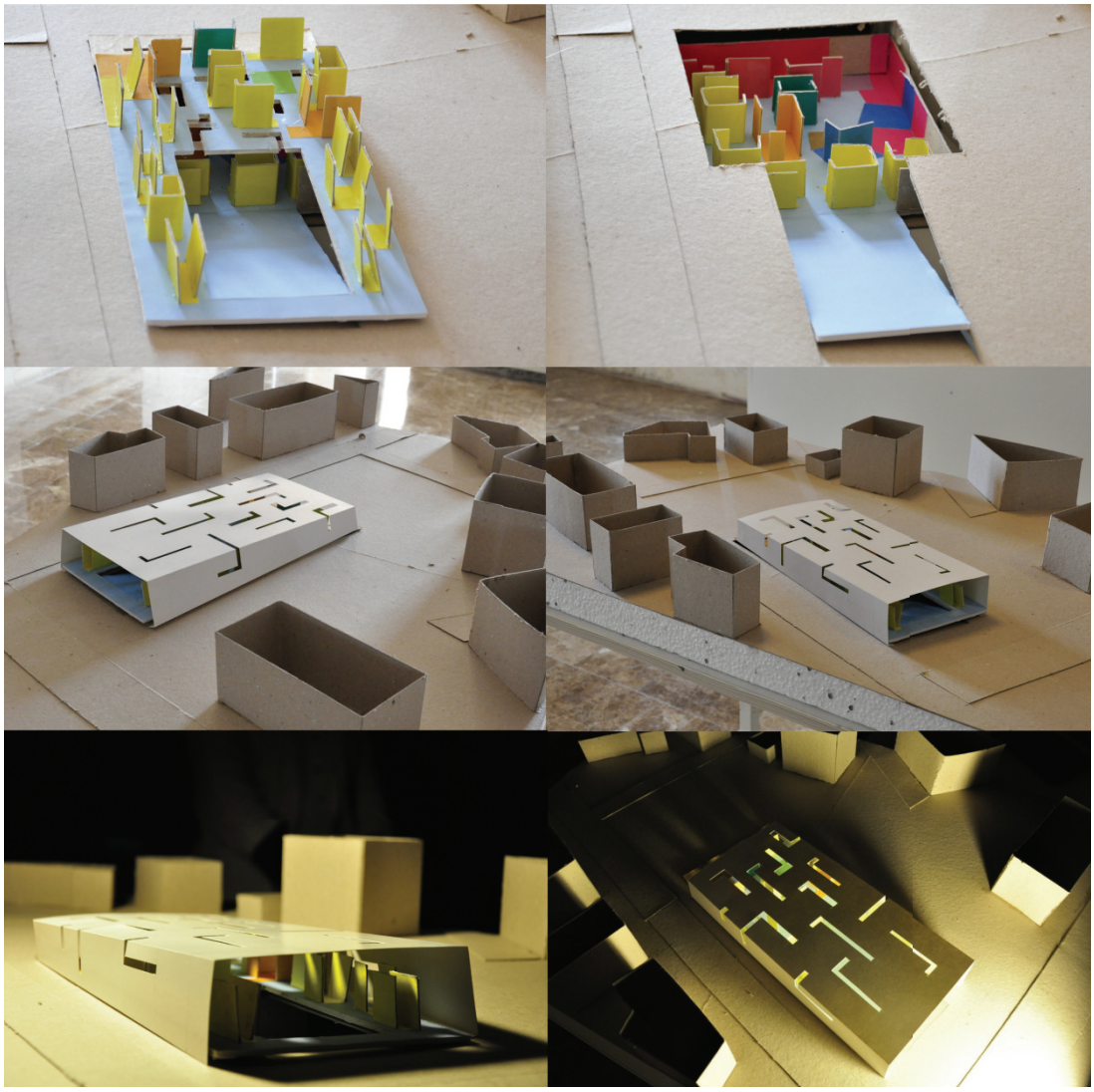


Over the circulation spaces that were left between DUs, the team started opening floor rifts, which were to be bridged by semi-transparent elements where necessary (Figure 19). The interior spaces were being connected both visually and spatially through both the large ramp and these rifts.

▼ Figure 19: Merve, Bahri, and Melike, sketches toward the final model.



The resulting building is dominated by the huge ramp (Figure 20). This could have been precluded or limited by the tutors or the designers in another context as irrational or excessive. However, in the context of this workshop, it was rather costless to move forward and bring the proposal to a level of development before deciding on this issue. In just 2,5 days, the proposal has matured to such a level that it became possible to evaluate the connection between the initial ideas, three-dimensional spatial arrangement, functioning of the building, and contextual relations in combination. The building obtained an iconic quality within its surroundings, the conceptual narrative was consistent and strongly related with the resulting spatial drama, the formation and organization of the rifts were well balanced and unifying, and the dynamism of the functional spaces was reasonably organized. Most importantly, this was just the very beginning and if this were a continuing design process, this proposal would be a promising initial draft, which already includes many of the crucial architectural aspects to be further discussed, developed, and refined.



▲ Figure 20: Merve, Bahri, and Melike, final proposal.

Although these workshops did not assess the professional applications of the d_p.layout, it appears that such a workflow can be employed to quickly develop several initial ideas, just like an in-office competition. d_p.layout is highly flexible and it is mostly possible to creatively devise ways to define a wide range of layout elements and contextual properties in order to be able to handle new scenarios. This versatility eliminates the need for a constant redefinition of a large number of parameters. There are just a few required parameters, like DU width limits, whose redefinition could have been automated through the sizes and dimensions of the layouts.

d_p.layout accelerates the design process by making it easier to enter into layout generation. It is relatively easy for human designers to interpret

existing proposals like draft layouts. Compare this with a blank slate that has to be somehow structured, shaped, or filled; which can be seen as an unstructured situation, an open problem with no structure, with no path to move forward. Whenever human designers face draft proposals, it becomes easier for them to move forward by interpreting and refining these; the problem gains structure and the designers are set on a path. This makes the process both easier and more pleasurable. Therefore, an assistant who is capable of bringing forward rough proposals, although inferior, would have a utility within the design process.

8. CONCLUSIONS AND FUTURE RECOMMENDATIONS

This paper demonstrated a novel application of similarity-based evaluation through an evolutionary layout design assistant. The functioning of the approach is described and verified, the versatility of the resulting assistant, the intuitive graphical interface, and the usage scenario are demonstrated.

The resulting evolutionary assistant is able to be quickly adapted to a range of problems, none of which is strictly implied within the initial framework or task definition. Basing evaluations on a pool of existing designs or quickly delivered graphical specifications enables an EA to become an open-ended system. This way, apparently strict problem definitions have been kept open to fast updating, which should be compared to rather static approaches like adjacency matrices, lists of constraints, and fixed objective functions. Moreover, same approach can be extended towards adjacency matrices and design constraints, by extracting these from existing buildings, which would convert them into dynamic approaches.

The approach gives way to a simple interface; preference definition becomes a simple task as drawing out graphical elements and choosing examples as targets. Numerical parameter definition is out of question for these aims, because designers are not usually able to explicitly state what they know [32]. Moreover, an explosion of parameters arises when complicated problems like architecture are encountered.

As has been shown with the usage case, d_p.layout can be put to use in the very beginning of a design process where the problem is rather vague and the information is sparse. At this point, it can assume the burden of analyzing existing buildings. Using this extracted information, it generates flexible drafts that are open to development and enables its user to jump over these initial steps directly into design refinement with a less painful progression, which allows the determination of the problem in many aspects simultaneously, through fast but detailed probes into the problem. In the hands of its smart users, d_p.layout turns into an idea generator. If it could enter design practice from this point, this could open the way for its further development, towards a higher level of intelligence.

A primary issue is the constitution and utilization of target pools. It is

possible to share and reuse evaluative information, in the form of example pools. Each worked-out example can be tagged in terms of its typology, scale, and contextual properties and shared over an open platform. Currently, the approach is dependent on well-established typologies like libraries, schools, and museums. Inclusion of other typologies and the extension of each typology with more buildings are straightforward requirements, but are time-consuming tasks. Moreover, the determination of typologies is not straightforward in all cases. The problem also extends to sub-typologies, hybrid cases, and situations where no definite typology exists. These are open questions for the approach. Hybrid methods that comprise manual labeling, datamining, and clustering analysis may be investigated for potential solutions.

Even when a definite typology is detected, there might not be enough information for a specific DU type in a target building, which results in rather random placements. This shows a requirement for methods that would enable the collective usage of a series of targets together, as is the case with DU distribution to floors. Finally, more refined evaluation functions may be developed through machine-learning and statistical methods.

REFERENCES

1. Eastman, C. M., Automated space planning, *Artificial Intelligence*, 4, 1973, pp. 41-64.
2. Akin, Ö., Dave, B. and Pithavadian, S., Heuristic generation of layouts (HeGeL): based on a paradigm for problem structuring, *Environment and Planning B: Planning and Design* 19, pp. 33–59, 1992 doi:10.1068/b190033.
3. Baykan, C. A. and Fox, M. S., Spatial synthesis by disjunctive constraint satisfaction, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Volume 11 / Issue 04 / September 1997, pp. 245-262.
4. Flemming, U. and Woodbury, R., Software Environment to Support Early Phases in Building Design (SEED): Overview, *Journal of Architectural Engineering*, 1(4), 1995, pp. 147–152.
5. Damski, J. C. and Gero, J. S., An evolutionary approach to generating constraint-based space layout topologies, in: R. Junge, ed., *CAADFutures 1997*, Kluwer, Dordrecht, pp. 855-864.
6. Jo, J. H. and Gero, J. S., Space layout planning using an evolutionary approach, *Artificial Intelligence in Engineering*, 12 (1998) pp. 149-162.
7. Gero, J. S. and Kazakov, V. A., Evolving design genes in space layout planning problems, *Artificial Intelligence in Engineering*, 12 (1998), pp. 163-176.
8. Rosenman, M.A., The Generation of Form Using an Evolutionary Approach, in: Gero, J.S., Sudweeks, F., eds., *Artificial Intelligence in Design '96*, Springer Netherlands, 1996, pp. 643–662.
9. Rosenman, M. A. and Saunders, R., Self-regulatory hierarchical coevolution, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 2003, 17, pp. 273–285.
10. Michalek, J., Choudhary, R. and Papalambros, P., Architectural layout design optimization, *Engineering Optimization* 34, 2002, pp. 461–484, doi:10.1080/03052150214016.

11. Wong, S. S. Y. and Chan, K. C. C., EvoArch: An evolutionary algorithm for architectural layout design, *Computer-Aided Design* 41(9), 2009, pp. 649-667.

12. Inoue, M. and Takagi, H., EMO-based architectural room floor planning, *Systems, Man and Cybernetics, 2009*. SMC 2009, IEEE International Conference on. IEEE, pp. 518–523.

13. Flack, W. J. and Ross, B. J., Evolution of Architectural Floor Plans, in: Esparcia-Alcázar, A. I., ed., *Applications of Evolutionary Computation*, 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013.

14. Rodrigues, E., Gaspar, A.R. and Gomes, Á., An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique, *Automation in Construction* 35, 2013, pp. 482–498, doi:10.1016/j.autcon.2013.06.005.

15. Rodrigues, E., Gaspar, A.R. and Gomes, Á., An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology, *Computer-Aided Design* 45, 2013, pp. 887–897, doi:10.1016/j.cad.2013.01.001.

16. Rodrigues, E., Gaspar, A.R. and Gomes, Á., An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests, *Computer-Aided Design* 45, 2013, pp. 898–910, doi:10.1016/j.cad.2013.01.003.

17. Schön, D. A., *The Reflective Practitioner*, Basic Books, New York, 1983.

18. Hanna, S., Automated representation of style by feature space archetypes: distinguishing spatial styles from generative rules, *International Journal of Architectural Computing*, issue 01, volume 05, 2005.

19. Hanna, S., Representing style by feature space archetypes, description and emulation of spatial styles in an architectural context, in J.S. Gero, ed., *Design Computing and Cognition'06*, 2006, pp. 3–22.

20. Hanna, S., Defining implicit objective functions for design problems. *GECCO'07*, July 7–11, 2007.

21. Wiens, A.L. and Ross, B.J., Gentropy: evolutionary 2D texture generation, *Computers and Graphics*, 26(1): pp. 75–88, 2002.

22. Wittgenstein, L., *Philosophical Investigations*, Translated by G. E. M. Anscombe, Blackwell Publishers, 1999.

23. Dreyfus, H. L., *What Computers Can't Do: A Critique of Artificial Reason*, Harper & Row, First edition, 1972.

24. Lakoff, G. and Johnson, M., *Philosophy in the flesh: the embodied mind and its challenge to Western thought*, Basic Books, New York, 1999.

25. Wittgenstein, L., *Tractatus Logico-Philosophicus*. Routledge, London; New York, 2003.

26. Dreyfus, H. L., *What Computers Still Can't Do: A Critique of Artificial Reason*, The MIT Press, Cambridge, Massachusetts, London, England, sixth edition, 1999.

27. Dreyfus, H. L., Dreyfus, S. E. and Athanasiou, T., *Mind over machine: the power of human intuition and expertise in the era of the computer*, Free Press, New York, 1986.

28. Zitzler, E., Laumanns, M., and Bleuler, S., A tutorial on evolutionary multiobjective optimization, in: Gandibleux et al., eds., *Metaheuristics for Multiobjective Optimisation*, Lecture Notes in Economics and Mathematical Systems, Volume 535, 2004, pp 3-37.

29. Sönmez, N. O., Erdem, A., and Sarıyıldız, I. S., Automated Evaluation and Generation of Graphic Arrangements through Adaptive Evolution, *Generative Art 2010*, Milano.

30. Michalewicz, Z. and Schmidt, M., Parameter Control in Practice, in: Lobo, F. J., Lima, C. F., and Michalewicz, Z., eds., *Parameter Setting in Evolutionary Algorithms*, Series: Studies in Computational Intelligence, Vol. 54, XII, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 277-294.
31. Smith, J. R. and Chang, S. F., Querying by color regions using the visualseek content-based visual query system, *Multimedia Systems*, 7: pp. 129–140, 1999.
32. Niedderer, K., Mapping the Meaning of Knowledge in Design Research, *Design Research Quarterly*, 2 (2), April 2007.

N. Onur Sönmez

İstanbul Technical University, Faculty of Architecture
ITU Faculty of Architecture, Taskisla Building, Beyoglu, Istanbul, TURKEY

onursonmezn@yahoo.com