Algorithmic Support of Creative Architectural Design

Tomor Elezkurtaj and Georg Franck

Vienna University of Technology, Department of Computer Aided Planning and Architecture, Treitelstrasse, A-1040 Vienna, Austria

e-mail: tomor@iemar.tuwien.ac.at *or* franck@iemar.tuwien.ac.at

Abstract

The system discussed in this paper applies artificial evolution methods in architectural floor plan design. Design variants are treated as copies of an evolving species. The properties that are subject to evolution pertain to the form as well as the function of the rooms accommodated. Since function in architecture is an open concept that not only includes physical comfort but aesthetic aspects as well, the system is not intended make the evolution of designs automatic. Rather, it invites the user to intervene interactively into the process as displayed on the screen.

The Problem of Algorithmic Support of Creative Architectural Design

After decades of research, computer aided architectural design (CAAD) still waits to be supported by methods of artificial intelligence (AI). The standard functionality of commercial CAAD software is confined to drawing, construction and presentation: in architectural design, computers are used to handle complex geometries, sophisticated visualization and performance simulation. The core business of architecture, the shaping of spaces and

organization of functions, is still not considerably supported by computerized methods.

The great expectations of the CAAD pioneers were fuelled by the impressive early progress of symbolic AI. The approach of symbolic AI to human intelligence is that of programming the use of language. Language is a broad concept, encompassing the use of words, symbols and even shapes. The way suggesting itself of combining CAAD with symbolic AI is formalizing shape grammars. Shape grammars are sets of forms, symbols and rules defining the way in which, e.g., meaningful architectural plans are composed of elements representing walls, ceilings, windows, doors, stairs etc. Plans are meaningful if they are well formed, i.e., if the elements are defined in a clear-cut way and manipulated according to syntactical rules. The art of programming a computer to construct architectural plans by manipulating symbols lies in formalizing the grammar and specifying its rules in such a way that syntactically well-formed expressions are architecturally well done as well.

Remarkably, the use of computer driven shape grammars came close to passing an architectural Turing test. Computerized grammars generated drawings of mock Palladian villas and fake Frank Lloyd Wright Prairie Houses that could easily fool even the expert eye if presented as long-forgotten originals.[1] Nonetheless, symbolic AI never came up with modules suitable for commercial CAAD software. The reason is that architectural design cannot be reduced to producing graphics and imitating style. Architectural plans have to be not only syntactically well-formed, but also meaningful. The meaning of architectural plans lies in the function that the objects represented are supposed to fulfil.

---

[1] Mitchell, W. The Logic of Architecture (MIT Press, 1990, p. 179)

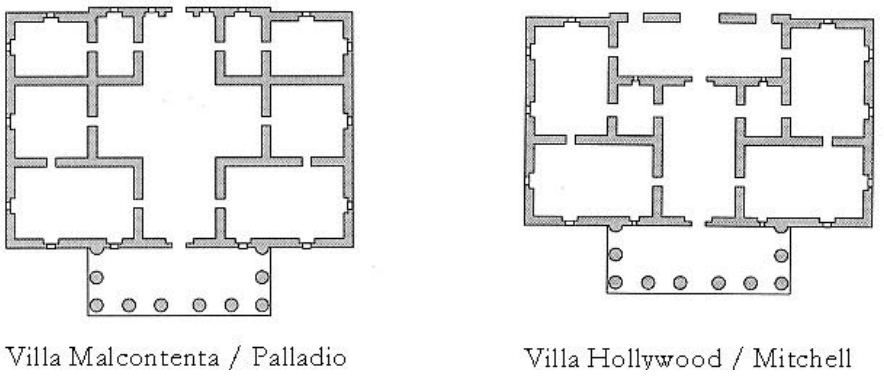

Figure 1: Shape grammar of Palladian villas. Right: original (Villa Malcontenta), left: a floor plan “invented” by the machine (Villa Hollywood by Mitchell / Stiny.

Function, however, is a broad and elusive concept in architecture. The function of a building lies in serving the needs and wants of its users and beholders. The definition of function thus faces two basic difficulties. First, it is unclear who counts as a user or a beholder. Second, the needs and wants are subjective in nature; the concept of function is no less subjective than that of beauty. Moreover, the needs and wants to be accounted for include those of the aesthetic sense. The function of a building (as well as the functions of its constituent parts) cannot be reduced to technical requirements.

Nonetheless, some functional requirements can be operationalized. A good deal of the function of the rooms in a building can be expressed in terms of neighborhood relations. The characteristic function of a hall, e.g., lies in giving access to the rest of the rooms that make up the house or dwelling. The prescriptive meaning of names like “kitchen” or “bathroom” lies to a great extent in the constraints and preferences pertaining to the shape and placement of the rooms in the context. Hence, it is good practice to start working on a plan by translating the names of the rooms listed in the program into preferred size, shape, orientation and neighborhood relations. This translation is the first step to dealing with function algorithmically.

Translating function this way into tangible criteria does not solve all the problems with functions. One should not disregard the difference between the full-fledged subjective concept of function and the concept defined implicitly by the tangible criteria. Neglecting this difference would reduce architecture to engineering. The design of buildings becomes an art precisely through dealing with the subjective needs and wants of the users and beholders. Our sensitivity to architectural qualities goes further than our ability to describe the needs and wants in explicit terms.

Because of this, the attention to function in the unrestricted sense is not only a source of risk and labor, but also a source of surprise. Being able to surprise means to be creative. Architecture is an art in that it offers the opportunity of satisfying more needs and wants than the users and beholders were conscious of having.

Algorithms that support creative design should help to produce surprises or things that one cannot foresee. How can algorithms be made to produce things not foreseen? What is needed is randomness. Of course, it would not be enough to randomly generate shapes or combinations of shapes. In order to be helpful, the play with chance has to be combined with goal-oriented search. It has to be turned into an effective, if not efficient, way of exploring a search space.

The problem of exploring the search space of architectural design efficiently is that it is hard to demarcate that space. The search space escapes being circumscribed in a conclusive way as long as function remains a loosely defined concept. In principle, it is impossible to deal with ill-defined problems algorithmically. What seems to be possible, however, is a division of labor between the human designer who is equipped to deal with ill-defined problems and the machine that is not.

The application presented in this paper draws a line between the search space demarcated by tangible criteria of functionality and the space of solutions that satisfy more subtle conditions as well. Since the tangible criteria are open to parametric variation, it may be reasonably assumed that the space of

interesting functional solutions is contained in the space of solutions that fulfil the tangible conditions.

Artificial Evolution

One of the most basic tasks for an architect is floor plan design. Floor plans are the conventional and proven means of accommodating a program in a situation determined by the shape and size of the lot, the zoning and building regulations, neighborhood and access relations and other factors.

A system supporting creative architectural design in the sense characterized above should display the following features:

1. It should provide an interface for interactively specifying programs and composing floor plan layouts.
2. It should account for the function of rooms in terms of (a) the size, shape, and orientation preferred and (b) in terms of constraints and preferences pertaining to neighborhood relations.
3. It should search the space of solutions to the placement problem in a way that optimizes the preferences specified.
4. It should it make easy (a) to intervene into the search process and (b) to change the values of the criteria specified.

If we also add the requirement that the system should produce surprise, we find ourselves asking a certain kind of creativity of the system itself. Even though creativity may seem too much to be asked of an algorithm, there are processes in nature that are both creative and capable of being described algorithmically: most notably, the strategies which biological evolution employs on the genetic level. The way genomes are selected, recombined and reproduced in the evolution of a species can be described in algorithmic terms as well as simulated to a certain degree by the computer.

The application of strategies discovered in biological contexts for problem solving and optimization forms the essence of what has been called New AI.

New AI differs from the old, symbolic, AI in that the paradigm of intelligent behavior has shifted. Instead of taking language as the paradigm case of intelligence, the focus is now on the adaptive behavior of organisms or populations of organisms.

Among the strategies that proved to be surprisingly powerful in problem solving are Evolution Strategies (ES) and Genetic Algorithms (GA). Both ES and GA simulate what is happening in the gene pool of a population of individuals competing for survival. The individuals that ES and GA act upon are the candidate solutions of design variants. The gene pool consists of the selection of the bit strings that encode the variants. The difference between ES and GA lies in the role that randomness plays in the process of selection and reproduction. In ES, randomness is confined to mutation, i.e., changing single genes. In GA, both mutation and crossover take place. Crossover means that individuals are selected for mating and recombining their genomes. In GA, both the couples that mate and the location in the genomes where they are crossed over are selected randomly. Selection according to the fitness of the individual variants takes place in reproduction. The fitter the variant, the higher is its chance to reproduce. A more detailed description of ES and GA is given in the appendix below.

ES and GA have proved useful in a wide range of placement problems.[2] They have become standard approaches to cutting and packing problems[3], they are routinely used in VLCS chip design[4]. There are a lot of papers on coding hybrid strategies and there are applications even using them interactively. There is a conspicuous lack, however, of applications that would help the designer of architectural plans and satisfy the requirements (1) – (4) listed above.

---

[2] E. Goodman, A. Tetelbaum, V. Kureichik A Genetic Algorithm Approach to Compaction, Bin Packing, and Nesting Problems (1994)

[3] See, e.g., Franck et al. (1999).

[4] Schnecke, V., Vornberger, O., An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization (1996) Parallel Problem Solving from Nature -- PPSN IV

In the remainder of this paper, an implementation of requirements (1) – (4) is presented. The system is made of three parts. The first part is an Evolution Strategy (ES) that fits rooms of given size and preferred shape into the outline of a building or, for that matter, storey. The second part consists of a Genetic Algorithm (GA) whose task is to arrange the rooms according to functional requirements that are expressed in terms of neighborhood relations. The third part is the user interface on which the design variant achieving the best score at the time is displayed and made accessible to intervention via the mouse.

The task of fitting a number of rooms of given size and preferred shape into an outline consists of finding out the arrangement that optimizes the proportions preferred and minimizes gaps, overlaps and overflow. Characteristic of the search space of this particular problem is a multitude of global optima. The risk of being caught in a local optimum is negligible. Since the space to be worked through is nevertheless ample, a simple ES is adopted for reasons of speed.

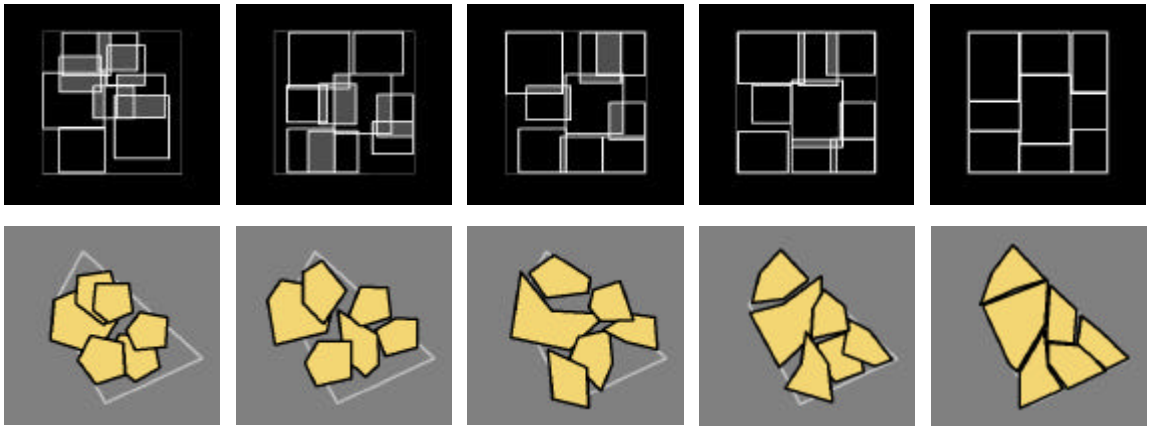

Figure 2: Fitting rooms of a given size and a preferred shape into an outline by minimizing gaps, overlap and overflow (as performed by an Evolution Strategy)

Fitting rooms into an outline is a simple task compared to the overall optimization of the neighborhood relations between them. The search of this optimization problem is much more complex, and the searchspace needs to be worked through more thoroughly. In order to accomplish this demanding task, a GA combining mutation with crossover is adopted. The operation performed by the GA is a kind of re-interpretation of the rooms arranged. It changes the functions attributed to the rooms in order to optimize the neighborhood relations. The output of the GA is thus turned into an input of the ES fitting in the rooms and vice versa (see Appendix).

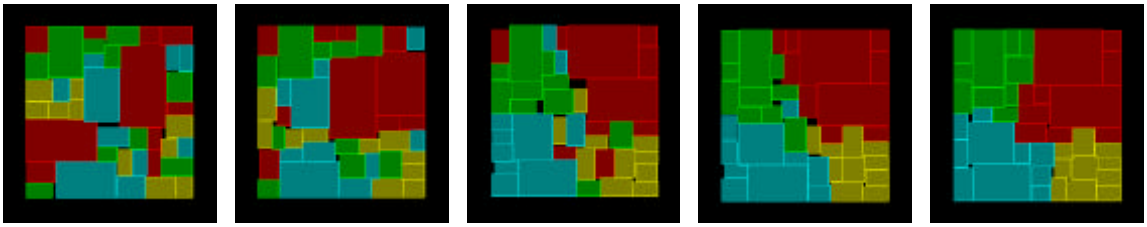

Figure 3: Rearranging the modules (parts of rooms) fitted into the outline according to the neighborhood relations preferred (by a Genetic Algorithm)

The reason for adopting this mixed strategy lies, among other things, in speeding up the process. Speed is crucial for interaction with the user.

The interface showing the design variant with the highest score at the time provides, at the same time, the possibility of changing the position, orientation, shape and size of rooms during the run. Moreover, it allowes changing the outline into which the rooms are to be fitted, the number of rooms to be accommodated, and the weights of the neighborhood relations between them (see figure 4).

Interaction is crucial for involving the capabilities of the designer that are out of reach for the machine. It can be assumed that human intelligence or intuition can deal with the complexity and opacity of the needs and wants entailed by the broader concept of function. Moreover, the human designer is used to exploit the gap between the intuitive intelligence of perception and the ability to describe what is perceived. This gap is the source of qualities that cannot be

made, but have to be found. Properties that cannot be planned prospectively, but appreciated in retrospect are what lie at the base of architectural quality[5] The best a machine can do in order to let serendipitous qualities emerge is to engender surprise.

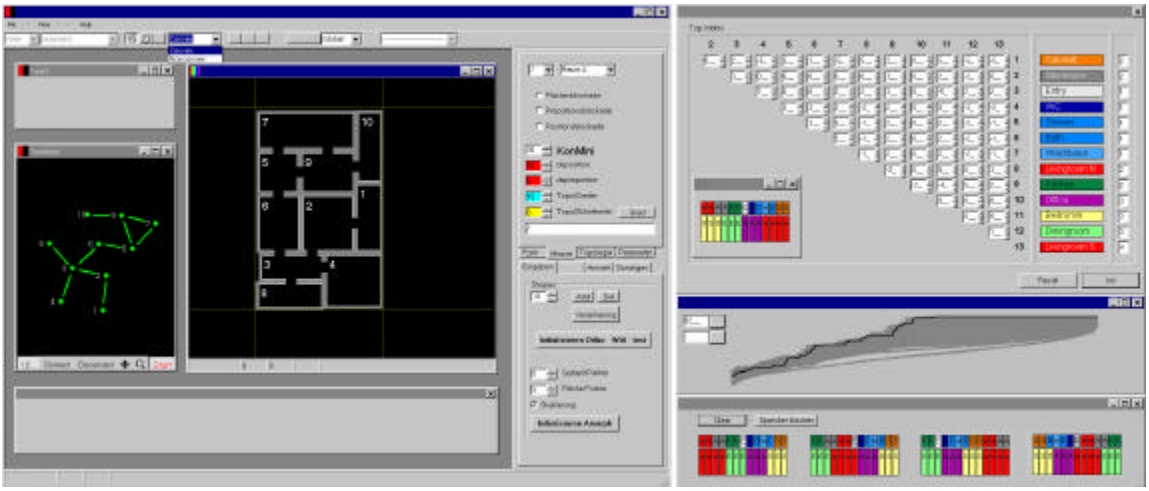

Figure 4: User interface displaying the topology (access relations) left; geometry (floor plan) middle; the topological matrix (containing the relative weights of the neighborhood relations preferred)

Artificial evolution has the potential to produce surprise. It is a method of accessing problems that resist being described analytically. Describing problems analytically means to describe them in a way that demonstrates that they can be solved and even implies how this may be achieved. The shape grammars of symbolic AI proceeded that way. The production and reduction (parsing) rules they work with are specified in a way that whatever solution is syntactically well-formed looks architecturally well formed as well. The application of ES and GA circumvents this prescriptive anticipation. In order to put ES and GA to work, only the objective function and the material to be worked with need needs to be specified. Of course, the specification needed is far from trivial. What is particular, however, is that ES and GA make use of randomness to an extent that prevents an exact prediction of the result. As a rule, it is not even possible to prove that the solution they render as optimal is

[5] On this point see Franck / Franck (2001).

indeed an optimum. The idea behind the system presented was that this vice can be turned into a virtue if the designs produced with its help interactively are susceptible to assuming the property of being not predictable but eye opening in retrospect.

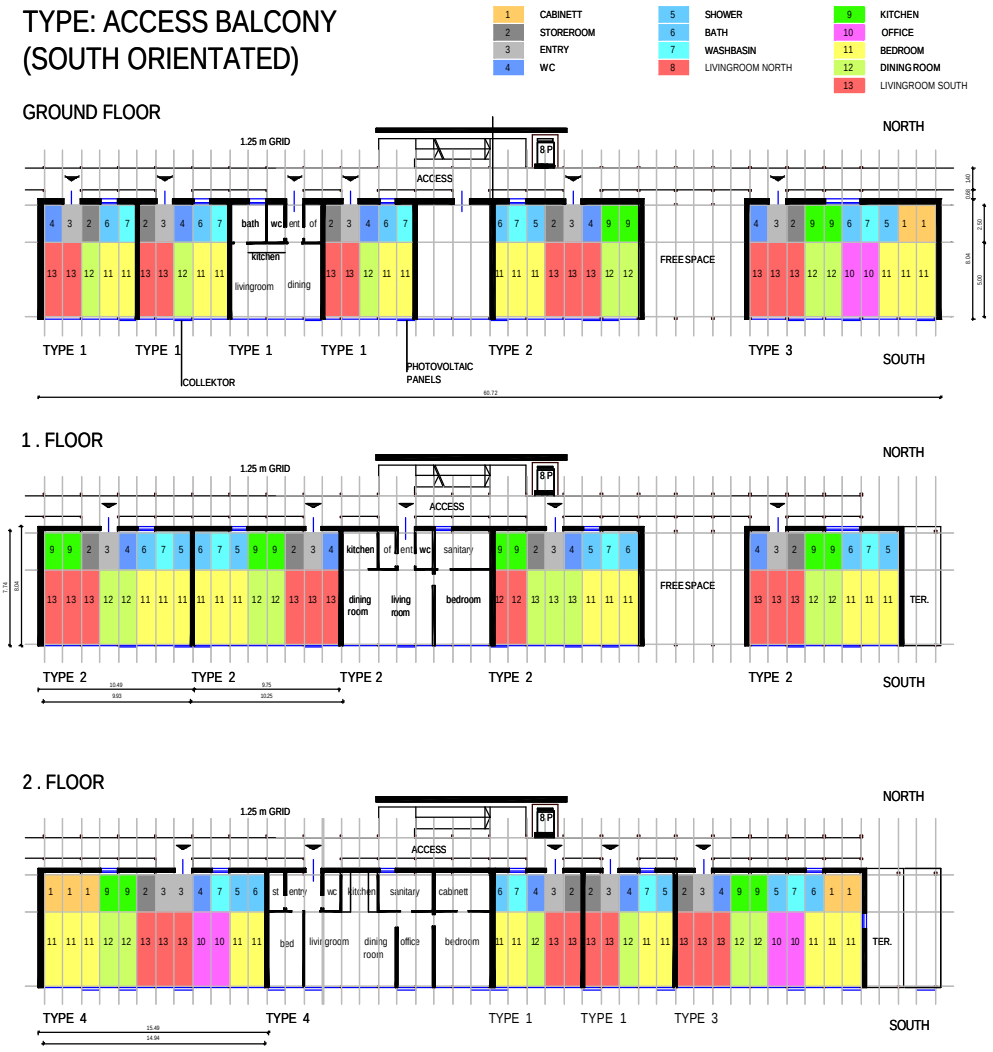

Figure 5: Design studio on ecological housing. Project by Christian Kadletz

The system of turning artificial evolution into a design medium has proved its potential to produce surprise in a series of design studios at the TU Wien. In a studio on ecological housing, houses in low-cost wood construction were developed that show an enormous flexibility and variability of uses (for an example see figure 5). In a design studio on urban redevelopment, the system was used for visualizing the sensitivity that plans optimizing neighborhood

relations show to minor changes in the boundary conditions or in the values attributed to the relations (for a series of variants resulting from minimal changes see figure 6). Even though these tests made clear that it will be difficult to develop an intuitively satisfactory user interface, they showed a great potential of facilitating unexpected results.

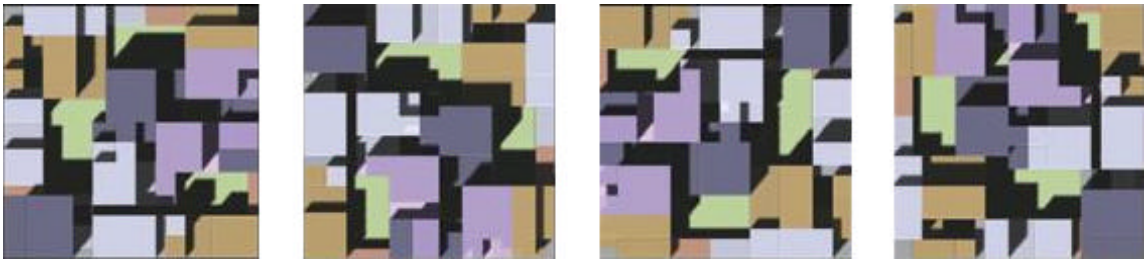

Figure 6: Urban redevelopment project. Variants by Christian Hoffmann

Outlook

The system developed so far is a prototype that demonstrates the feasibility of the approach. The steps of development now under work go into three directions. First, the geometry of the forms that the system is suited to handle waits to be further adapted to the requirements of architectural and urban design. The forms that can be handled by now are rectangles and higher polygons. Since polygons of higher order increase the complexity enormously and thus considerably lower the performance of the system, a more efficient way of handling free forms and complex geometries is needed. Second, many buildings have more than just one storey. The system thus waits to be generalized concerning the number of storeys dealt with at a time. The approach adopted lies in exchanging the topological matrix (see Appendix) for a 3D tensor that accounts for the neighborhood relations in the vertical direction, as well. Again, the problem is that of increasing complexity. Third, the criteria of functionality entering the objective (or fitness) function wait to be enhanced. This can be done on the basis of data that result from simulating the

performance of buildings. It can be argued that performance simulation and artificial evolution should be made to collaborate.

Appendix

A. Evolution Strategy

ES was developed in the early sixties by Ingo Rechenberg[6] and Hans Peter Schwefel[7]. In its original form, real-coded parameters were used and subjected to mutation. In contrast to GA, mutation, as search operator, acts on the phenotype or individual. The presupposition that the coding the variables in the ES starts from is that a sufficiently strong causality (small changes of the cause must create small changes of the effect) prevails.

The problem to be solved by the evolution strategy described in this paper is considered as a kind of a so-called Bin-Packing[8], Nesting or Cutting Problem[9]. For solving the geometry of the architectural floor plan, the fitness-function ($Sg \rightarrow 0$) of the ES minimizes the sum total of overlap between the shapes *(ij)* to be accommodated and the outside area *(u)*.

$$Sg = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (S[i] \cap S[j]) + \lambda \sum_{i=1}^{n} (S[i] \setminus S[u])$$

After being initialized, a population of design variants is subject to random change concerning position and proportion. Selection acts through reproduction from generation to generation. The fitter a variant, the higher is its reproduction rate. The proportion preferred is approximated through filtering probabilities.

---

[6] Rechenberg, I., Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution , Frommann-Holzboog, Stuttgart, 1973

[7] Schwefel, H.P. Numerical optimization of computer models , Wiley, Chichester, 1981

[8] Falkenauer, E. A Hybrid Grouping Genetic Algorithm for Bin Packing (1996)

[9] Hinterding, Juliff , A genetic algorithm for stock cutting (1993)

B. Genetic Algorithms

The study of genetic algorithms originated with John Holland[10] in the mid-seventies. A genetic algorithm is an iterative procedure that consists of a population of individuals, each one represented by string of symbols, encoding a possible solution to a given problem. Offspring is generated by way of crossover and mutation. Individuals are selected for recombination with a probability proportional to their relative fitness.

The genetic algorithm described in this paper solves the topology of the architectural floor plan. The problem to be solved is a so-called Quadratic Assignment Problem[11] (QAP) or a facility placement problem. Consider the problem of allocating a set of facilities to a set of locations, with the cost being a function of the distance and flow between the facilities. The objective is to assign each facility to a location such that the total cost is minimized. Translated to the topological problem of the floor plan, the locations are the rooms (walls), the facilities are the preferences of the rooms (size, proportions etc.) and the flow is expressed as the importance of the neighborhood relations between the rooms. There are two matrices with the elements W = (w_{ij}) and T = (t_{ij}). The elements w_{ij} of matrix W express the importance of the neighborhood between room (i) and room (j), the elements t_{ij} of the matrix T represent the distance between the location (i) and location (j). The weights n_{ij} are specified by the user (see figure 4).

The value of n_{ij} expresses the importance of the relationship between rooms i and j. $n_{ij} = 0$ means that the distance is without importance, $n_{ij} < 0$ means that the rooms should be as distant as possible from each other. The fitness function ($W_T \rightarrow \min$) is to minimize the products $w_{ij} * t_{ij}$.

$$W_T = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(w_{ij} * t_{ij} \right)$$

[10] Holland J.H., Adaptation in natural and artificial system, Ann Arbor, The University of Michigan Press, 1975

[11] Burkard, Rainer E. The Quadratic Assignment Problem (1998)

References

Elezkurtaj, Tomor (2000), Evolutionary Algorithms in Support of Architectural Ground Plan Design, in: Art, Technology, Consciousness, ed. by Roy Ascott. Intellect Verlag. ISBN 1-84150-041-0

Elezkurtaj, Tomor / Georg Franck (1999), Genetic algorithms in support of creative architectural design, in: eCAADe17, Architectural Computing. Proceedings of the 17th Conference on Education in Computer Aided Architectural Design in Europe, ed. by André Brown, Michael Knight and Philip Berridge, Liverpool 1999, pp. 645-51

Franck, Dorothea / Georg Franck (2001), Qualität. Von der poetischen Kraft der Architektur, in: Merkur Nr. 626 (Juni 2001), S. 467-80. A translation into English is available under: http://www.iemar.tuwien.ac.at/modul23/text_pdf/Franck_2001d_arch_e.pdf

Franck, Georg / Dietmar B. Schweiger / Karl Svozil (1999), A packing problem, solved by genetic algorithms, in: Journal of Universal Computer Science, vol. 5, no. 8, pp. 464-70

Frazer, John (1995), An Evolutionary Architecture, London: Architectural Association

Goldberg, David E. (1989), Genetic Algorithms in Science, Optimization, and Machine Learning, Reading, Mass.: Addison Wesley

Holland, John H. (1995), Hidden Order, Reading, Mass.: Addison Wesley

Koning H./ J. Eizenberg (1981), The language of the prairie: Frank Lloyd Wright's prairie houses, in: Environment and Planning B 8, pp. 295-323

Mitchell, M. (1996), An Introduction to Genetic Algorithms, Cambridge, Mass.: MIT Press

Stiny, Georg/ William J. Mitchell (1981), The Palladian grammar, in: Environment and Planning B 5, no. 1, pp. 5-18