Genetic Algorithms in Support of Creative Architectural Design

Tomor Elezkurtaj and Georg Franck

The functions supported by commercial CAAD software are drawing, construction and presentation. Up to now few programs supporting the creative part of architectural problem solving have become available. The grand hopes of symbolic AI to program creative architectural design have been disappointing.

In the meantime, methods called referred to as New AI have become available. Such methods includegenetic algorithms (GA). But GA, though successfully applied in other fields of engineering, still waits to be applied broadly in architectural design. A main problem lies in defining function in architecture. It is much harder to define the function of a building than that of a machine. Without specifying the function of the artifact, the fitness function of the design variants participating in the survival game of artificial evolution remains undetermined.

It is impossible to fully specify the fitness function of architecture. The approach presented is one of circumventing a full specification through dividing labor between the GA software and its user. The fitness function of architectural ground plans is typically defined in terms only of the proportions of the room to be accommodated and certain topological relations between them. The rest is left to the human designer who interactively intervenes in the evolution game as displayed on the screen.

Keywords: *Genetic algorithms, creative architectural design*

A Brief History of AI in CAAD

The history of computer aided architectural design is not without irony. Starting with raising hopes of turning creative design into a scientifically disciplined method of problem solving [A], the computer eventually entered the architectural business as a down-to-earth means of saving costs. Instead of substituting the human designer, the computer proved a useful tool for drawing and constructing, a convenient mailbox and filing cabinet. Today, creative design is as intuitive, non-scientific and chaotic as it has ever been. The most conspicuous jobs done by computers in architecture are sophisticated presentation and on-line co-operation.

The high hopes of the early days were fuelled by the then impressive progresses 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, for example, meaningful architectural plans are composed of elements symbolizing walls, ceilings, windows, doors, stairs etc. Plans are meaningful only

if they are well formed, which means that the elements are defined in a clear-cut way and manipulated according to syntactical rules. Take a shape grammar rich enough for composing architectural plans, formalize it, program it, and you have enabled the machine to enter a trial-and-error process of producing plans which, in turn, are capable of being evaluated and selected in the manner candidate moves in chess are.

Remarkably, the use of computer driven shape grammars came close to passing the architectural Turing test. Computerized grammars came up with, for instance, mock Palladian villas and fake Frank LLoyd Wright prairie houses which it would be hard to identify as imitations if presented as long-forgotten originals [B]. Nonetheless, symbolic AI never came up with modules suitable for integration with commercial CAAD software. The reason is that design strategies promising in architecture resist being reduced to a calculus of winning.

The problem of winning a chess match is well defined. The problem of winning an architectural competition is ill defined. In chess, problem solving consists in scanning alternatives that are given. In architecture, problem solving may consist in adapting existing designs as well. Characteristically, however, creative solutions are those redefining the problem until a design favored for heterogeneous reasons arguably becomes a solution. In chess, redefining the problem to be solved is simply forbidden. In architecture, the problem to be solved is systematically open to revision since function in architecture is an open concept.

Everybody - except, perhaps, Peter Eisenman - agrees that architecture has to be functional. No one has ever succeeded, however, in describing the function of architecture exhaustively. As soon as the exercise of describing the function of a building is taken seriously, the description dissolves into an unending list of criteria. These criteria pertain to the needs and wants not only of the immediate users of the building, but to those of each and every one for whom the existence of the building has some positive or negative value. The question after the function of architecture is one which cannot - and must not - be finally settled. Without giving a complete description, however, of the function to be served by the architecture, the problem of architectural design remains ill defined and open to interpretation. Creative design turns this vice of ill-definedness into the chance of inventing things never seen before.

The strategy of problem solving pursued by symbolic AI was that of decomposing complex problems into simpler problems until a level of elementary problems is reached. This strategy proved successful in areas where two conditions are fulfilled. The first condition is that such a level of elementary problems in fact exists, the second condition is that the solutions of these elementary problems can be formalized. In creative architectural design, the first, and thus neither, of these conditions if fulfilled. Preparation of the design problems to be solved by manipulating shape grammars thus consisted in a radical re-interpretation of what architecture means. It consisted, that is, in consistently disregarding the functional aspect of architecture. The 'sentences' formed by the use of shape grammars only ever were *syntactically* well formed. The search strategies only ever looked for solutions satisfying the *formal* prerequisites for being a possibly meaningful plan. The meaning itself was elaborately kept out of the process. The designs emulating the famous examples did so by restricting themselves to the purely formal aspect of the shapes manipulated. The design strategies were successful because of, not in spite of, disregarding any pragmatic or semantic meaning of the designs produced.

Emulating a style of design is one thing. Helping the designer in being creative is another one. In the first case the emphasis is on reproduction, in the latter case it is on exploration. Exploration in architectural design is rarely purely formal. Whether or not it is consciously guided by functional viewpoints, it obeys functional criteria as long as it is supposed to be architectural design and not just graphics. Exploration in architectural design may very well include re-

interpretation of functional requirements; functional requirements, however, that are fixed and accepted can be disregarded at the cost only of turning exploration into an idle play of forms. In order to facilitate and not just inspire exploration, the software should be capable of obeying functional requirements to some extent.

As soon as function is involved, the strategy of solving the problem by way of its final analysis comes to an end. The functional description of a building and thus of its components never is complete. Without complete description, there is no final analysis. So how is it possible to circumvent the need for a final analysis without foregoing the powers of AI? Through new AI?

New AI differs from old, symbolic, AI in that the paradigm of intelligent behavior has shifted. Instead of human language, it is now biological life that provides the paradigm cases of intelligent strategies. The well-known products of new AI are neural networks and genetic algorithms. Neural networks reproduce some (simple) features of the working of nervous systems; genetic algorithms orchestrate some (simple) features of biological evolution.

The approach presented in the following makes use of the capabilities of genetic algorithms (GA). Like neural networks, genetic algorithms are even further away from understanding than good old symbolic AI. GA simulate a generative process which is explicitly supposed to be 'blind'. There is no understanding and thus no meaning whatsoever in the way in which artificial evolution works. Nevertheless, evolutionary processes are the most creative and inventive known. Though allegedly primitive in comparison with biological evolution, GA have proved capable problem solvers in various fields of engineering [C]. They wait to be applied significantly to architecture [D].

Artificial Evolution

Applying GA to design problems means treating design variants as members of a population of candidate solutions that compete for survival in a game of evolution. Evolution proceeds in two steps: reproduction and selection. Reproduction means members of the population to mate and generate offspring. Selection means that the rate of reproduction depends on the 'fitness' of the individual

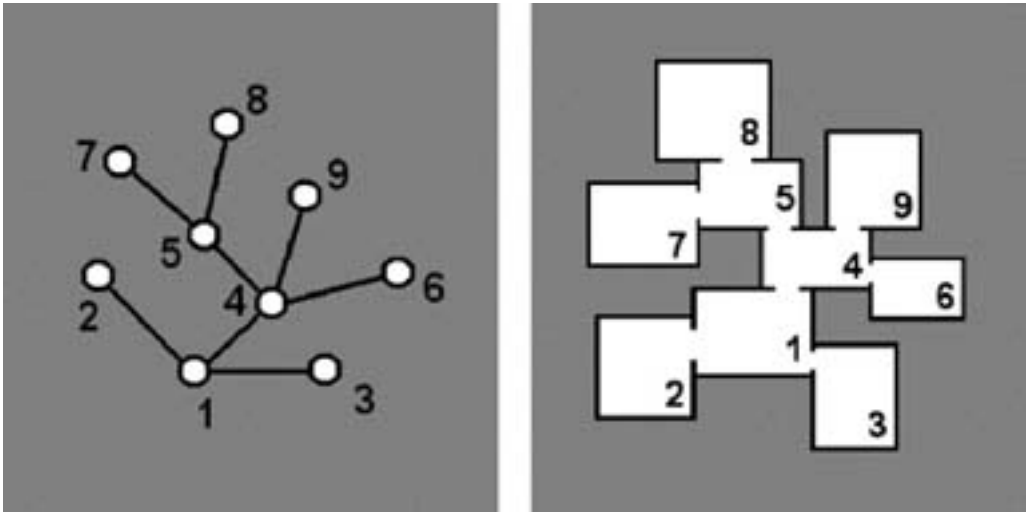

Figure 1 (right). Functional scheme and the size of rooms

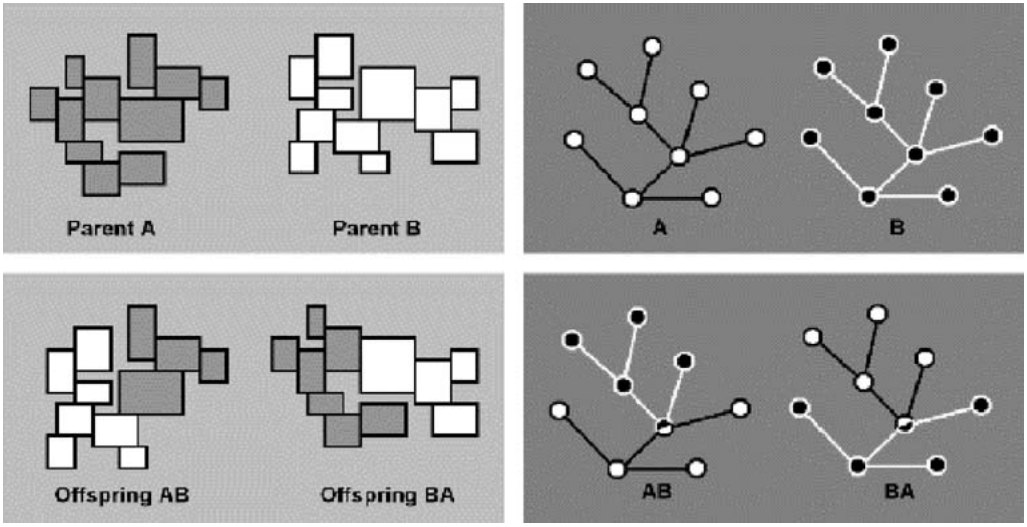

Figure 2 (left). Crossing over design variants

concerned. How do we make design variants mate? By treating the coding of their parts and properties as genes capable of being taken apart and recombined in certain ways. Mating is accomplished by *crossing over* the strings of bits encoding two 'parent' designs (see figure 2).

The reproduction process engendered by GA is 'blind' in that both the selection of the pairs that mate and the selection of the locations of the strings of bits at which the crossing over takes place are random. In addition to this two-fold kind of randomization, the strings are subject to a third kind of random change, acting in the manner mutation does in genetics. In the context of GA, mutation means that single bits are changed by chance from 1 to 0 or vice versa.

So far, reproduction is not only blind, but free also of any tendency towards optimality. However, reproduction turns into a means of optimization as soon, as selection acts on the strings reproduced. Selection, in the context of genetics, means that the individual *phenotypes* undergo a process of testing their 'fitness'. In biology, fitness is the capability of the individual to survive and reproduce. In artificial evolution, fitness measures some score of hits.

Putting aside for the moment the question of how the fitness of design variants is measured, let us see how GA combine selection with reproduction. The way in which selection acts on reproduction is straightforward: the higher the fitness, the higher the chance of reproduction. The features contributing to a high fitness thus proliferate in the population, those contributing below average tend to decrease with every new generation, those being of little help are doomed to vanish since individuals are subject to mortality. By iterating this game of reproduction and selection, the fitness of the relatively fittest individuals as well as the average fitness of the population is made to increase up to the point where an optimum is reached.

If fitness depends on more than a single variable, trade-offs are to be expected. Take the ground plan of a dwelling for illustration. The fitness of the ground plan depends, among other things, on (a) the proportion and orientation of the rooms and (b) on the neighborhood relations between them. A variant may be a good fit according to (a) and performing poorly according to (b). Since each variant is measured along both (a) and (b), the exploration of

the possibility space results in a 'fitness landscape' with the dimensions of a, b and fitness. In the dimension of fitness, this landscape will typically have hills and valleys, in which the hills are of different height. The hills represent optima; hills of heights different from the highest one represent local optima. The problem of optimization in such a fitness landscape is not just hill climbing, but finding out the global optimum. For this task, our model of mating and crossing-over is not yet equipped. It is just capable of hill climbing. It will be caught in a local optimum when such one is reached, thus ending up in a sub-optimal solution. In order to enable the optimization process to transverse valleys, another random process has to come into play: mutation.

The rate of mutation is the rate at which single bits in the strings are randomly changed from 1 to 0 or vice versa. Selecting an appropriate rate is critical for successful optimization. A rate too low prevents crossing the valleys; a rate too high tends to destroy the fitness accumulated. Interestingly, Mother Nature teaches us to select a rate of the order of 1:2.000. With a mutation rate of this order, optimization can be expected to be successful.

GA as a Tool of Architectural Design

The question of how to measure the fitness of a design variant inevitably brings us back to the problem of functional description. Asking how fit a building or a component is means asking how well it serves the function it is supposed to serving. If function resists being exhaustively described, the fitness of a design variant resists being measured in a reliable and reproducible way.

A way - if not the only way - of circumventing the need of a full specification of the fitness to be maximized lies in a division of labor between the programmer and the operator of the GA software. The operator or, for that matter, the designer is used to deal with ill-defined problems. A good deal of her or his intelligence just lies in the ability to act purposefully without having fully specified the aims set out. The gap of vagueness handled intuitively is one of the main sources of creativity. A system supporting creative design should be careful not to narrow down this gap prematurely.

Though there is vagueness in the definition of function in architecture, not all the functional requirements of a design are soft. There (usually) is a fixed size and location of the site, and there are regulations to be observed. Normally, there is a schedule listing the number and sizes of the rooms to be accommodated, and a functional scheme concerning the overall organization (see figure 1). Feeding these hard criteria into the system not only allows delegating a good deal of routine work to the machine, but to initialize also the artificial evolution of a ground plan.

In the application presented, initializing the evolution of ground plans proceeds in four steps. First, the outline of the building is specified. Second, the list of rooms to be fitted into the outline and the proportions preferred are entered. Third, the functional scheme of organization and access is specified. Finally, the game of artificial evolution is made to run. The system now tries to fit the rooms into the outline by (1) minimizing overlaps and gaps between the rooms, by (2) taking care the that the preferred proportions of the rooms are approximated, and by (3) preserving the functional scheme. It is this game in which the designer intervenes via the mouse (see figure 3).

The evolution, as displayed on the screen [E] now can be steered and controlled by relocating rooms freely within the plan, by changing their proportions and even sizes. In a subsequent or, for that matter, intermittent step, the outline and the list of rooms as well as the functional scheme can be altered. It becomes thus easy to check the consequences of changing the internal organization not only, but of varying the boundary condition as well.

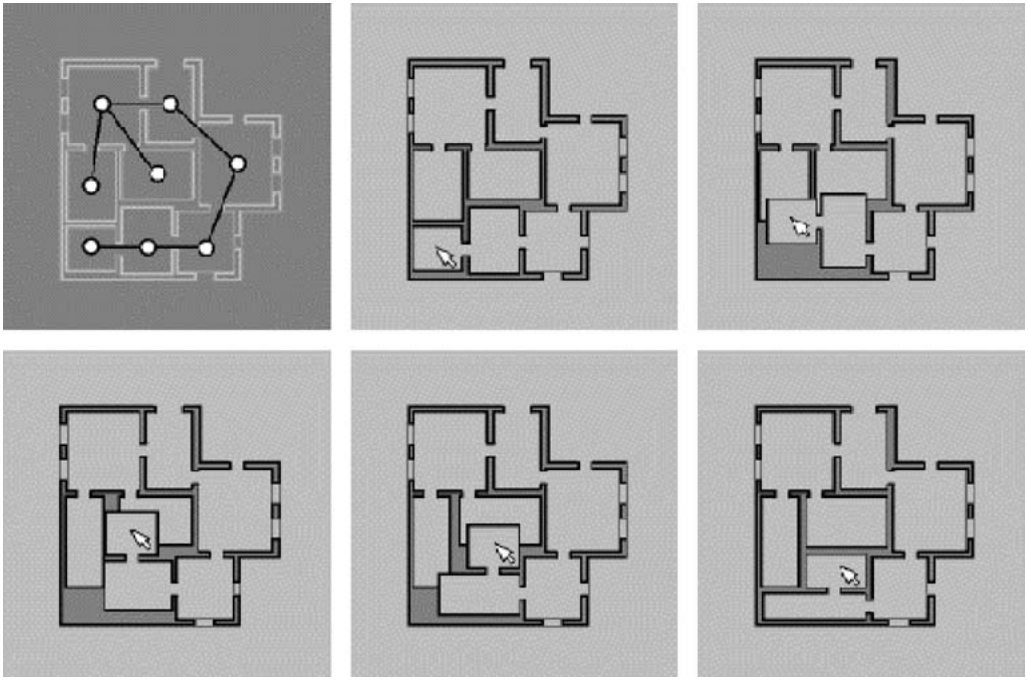

Figure 3 (left). Intervening in the evolution games via the mouse

Outlook

Currently, the system presented undergoes a first practical in a design studio held at the University of Technology in Vienna. The test concerns usefulness in general and handling in particular. The next major step in developing the system will be the transition from 2D plans to 3D models.

The system promises to be most useful in exploring fields of design where standard solutions have not yet crystallized. One such area is solar architecture. In order to subject the system to an enhanced test, a design studio is planned in the next winter term dealing with the problem of turning the zero-energy-house developed by Frauenhofer society in Freiburg, Germany, into a feasible type of a prefabricated house.

References

Alexander, Christopher (1964), Notes on the Synthesis of Form, Cambridge, Mass.: Harvard University Press

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

Franck, Georg/ Tomor Elezkurtaj (1999), Genetische Algorithmen als Entwurfshilfe der räumlichen Planung, in: Computergestützte Raumplanung. Beiträge zum Symposion CORP'99, ed. by Manfred Schrenk, Wien: Institut für EDV-gestützte Methoden in Architektur und Raumplanung. http://www.iemar.tuwien.ac.at

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

Notes

[A] See, e.g., Alexander (1964).
[B] See Koning/ Eizenberg (1981) and Stiny/ Mitchell (1981).
[C] As introductory texts see Goldberg (1989) and Mitchell (1996).
[D] For a general presentation of the idea and an overview of the approaches having surfaced until then see Frazer (1995).
[E] For a online presentation see: http://www.iemar.tuwien.ac.at/Forschung/Genetische_Algorithmen/genetische_algorithmen.html

Tomor Elezkurtaj and Georg Franck
Vienna University of Technology
Department of Computer Aided Planning and Architecture
Vienna, Austria
tomor@osiris.iemar.tuwien.ac.at
franck@osiris.iemar.tuwien.ac.at