Design Tools for Integrative Planning

Stefana Parascho¹, Marco Baur², Jan Knippers³, Achim Menges⁴
¹²University of Stuttgart, ¹ITKE, University of Stuttgart, ⁴ICD University of Stuttgart
¹s.parascho@itke.uni-stuttgart.de, ²marco@forcett.net, ³info@itke.uni-stuttgart.de,
⁴mail@icd.uni-stuttgart.de

Abstract. The performance of an architectural object is highly difficult to both define
and measure in its complexity since it is integrating a constantly increasing amount of
information, from concrete measurable characteristics to the subjective perception of
individual users. The question arising though is how to predict the performance of a
building and influence the design in order to increase it according to a significantly high
number of criteria.
The presented paper proposes two design tools, both developed and programmed in rhino
python for the generation of freeform geometries. The tools are generated for specific
tasks, but may be interpreted as exemplary for a way of defining and structuring a design
program in order to increase its efficiency. Both tools rely on a computational core that is
generally defined and may be fed with as many and different constraints and criteria as
considered suitable for the defined task.
Keywords. Integrative design; evolutionary algorithm; agent-based system.

INTRODUCTION
Both measuring and changing a design’s perfor-

mance is defined by such a high amount of informa-
tion that computational methods are of crucial im-
portance in this process. Computational tools may
help process a much bigger amount of information
and variables than one may be capable to capture
intuitively and through classical design methods.
The incorporation of digital tools should thus hap-
pen as early as possible in the design phase, even at
the point of analysing the given task. Still, using the
computational power of digital tools in architectural
design may prove itself more difficult than in other
industries, since a building is usually an individual
and very context-dependent object. In architecture
time and resources are often insufficient to allow
the development of highly performative designs
and design methods. Defining criteria of this perfor-
mance, for an architectural object is a process differ-
ent for each project, since context- and user-specific
factors vary constantly. Therefore generic programs
and tools can only cover a very general and unspe-
cific area of the planning process and may only help
in the representation and simplification of a design,
therefore being insufficient for capturing the com-
plexity of an architectural object.
Numerous established optimisation methods
have been incorporated into architectural design
opening up a new dimension of solution options
and a new freedom degree for designers, but at the
same time creating a new extremely complex prob-
lem as to how such tools are to be implemented and
further developed for the use in design tasks. Many
questions arise, such as which method is best suit-
able for architectural use, how this method is to be
translated and implemented, what the limitations are and how these are to be defined in order to increase the creativity of a design and not to define a too strict solution space limit.

Finally the paper proposes a comparison of two acknowledged optimisation tools and their implementation, pros and cons, for architectural design. The purpose is to modify and further develop the known tools and optimisation methods as to find an optimum implementation for specific design tasks.

STATE OF THE ART

**Bionics**
The traditional top-down design process has already been questioned and a number of bottom-up strategies have been developed and implemented in architectural design having as purpose to include a higher complexity into built design and create more performative and properties-specific results. Bionic processes are one of the main research areas in this development since they draw a parallel between the complexity of nature and architectural building systems. Looking at natural models as examples for specific characteristics and assets of these systems and implementing these into architecture through an abstraction of its principles leads to a differently hierarchized design process and a bottom-up method that includes more information into the early stages of design (Knippers and Speck, 2012). In these processes a specific natural phenomenon (e.g. self-organization), a characteristic of a biological system or a whole biological process (e.g. Evolution), is analysed and translated into design principles.

**Evolutionary Algorithms**
Evolutionary algorithms are widely used metaheuristic optimization algorithms developed in computer science and mathematics for problem solving (Ashlock, 2006). They are, similar to hill-climbing or simulated annealing, search algorithms meant to look through a solution space for the result of a complex optimisation problem. Developed also by John H. Holland (1995; 1998), as a reference to natural systems, evolutionary algorithms have been of major interest for architectural design from their first appearance in computer science. Their main strength, which designers are trying to make use of, is the capacity of multi-criteria optimisation, therefore searching for a solution that may fulfil more than one chosen criterion. At the same time following the natural system as a model, such optimisation methods show the immense potential that the natural systems have if used as examples in design processes.

The use of evolutionary algorithms has increased in architecture in order to achieve an optimisation of desired criteria, starting with Frazer J.H. in 1995. The final purpose was that their high potential for the built environment should be made use of, since they describe a much more complex system, with similarities to the natural one (Frazer, 1995)

**Agent-based systems**
A further development of evolutionary theory is the theory of self-organization, both meant to describe and explain complex and chaotic natural systems (Frazer, 1995). The main principle of self-organization implies that an organized system evolves out of a chaotic one only through the interaction of its parts and subsystems, without any higher controlling entity (Camazine et al., 2001). Agent models rely on the definition of a global system through its simple part, a so-called agent. These agents act according to a set of given simple rules, interacting with the other agents and developing complexity and emergent behaviour in the system (Reynolds, 1999) [1]. Examples of such systems in nature are swarms (flocks of birds, schools of fish or ant colonies) where all participating agents follow simple behavioural rules according to their neighbouring agents.

Although in computer science and mathematics agent-based systems are a widely used search algorithm for solving complex problems, in other fields it has been mostly used for simulation. In architecture agent-based systems have mainly offered solutions for crowd-simulation and circulation systems, ensuring an improvement of circulations and spatial arrangement, or particle simulation in animations. Few
research projects have used agent-based systems for the generation of form/geometry [2] while these systems often struggle with developing an autonomous non pre-defined system since one of the major difficulties in using agent-based systems is ensuring the convergence of the system in a working solution.

**DESIGN TOOLS**

**Evolutionary Design Strategies**

The first presented tool was designed for the generation of a high-rise project. Since the requirements for high-rise buildings present one of the most complex systems in building design, a specific design tool enabling an evaluation and improvement of the tower’s global performance was developed.

As intended, the design strategy is based on a computational core that allows multiple use and adaptation to a specific task. In this case a genetic algorithm was chosen for the core as a result of the numerous and complex requirements. A tower’s dimensions especially in height have as a result a high number of continuously changing criteria that need to be taken into account when designing a high-rise building. Thus the form of the building has often little flexibility and is very difficult to influence. While the presented project regards the outer form as the result of its inner constraints, the intention is to still have the possibility to choose the degree at which this form is influenced by the designer. At the same time numerous criteria, sometimes contradictory, make it very difficult to even only control these inner constraints so that the outer form often remains the pure functional result of these requirements.

While evolutionary algorithms show great results in optimisation problems with one criterion, their power for architectural design lies in handling more than one requirement and even working through contradictory optimisation criteria. The computational capacity of handling an incredibly high amount of information, comparing and changing this information, makes it very useful for such a complex task. While genetic algorithms can instrumentalise contradictory criteria, the result is not to be seen as an optimum to all chosen criteria but more as a compromise between the different criteria that depends a lot on the value of each criterion that is predefined by the user. It is very important to regard such optimisation results not like in mathematics as the singular and universal solution to a complex problem, but as the specific, task adapted and user dependent solution to one type of definition of the problem.

The process involved in using a genetic algorithm includes three steps: firstly the definition of a geometry generation algorithm, followed by the analysis after certain chosen criteria and the resulting fitness value, and lastly the recombination of the fittest individuals resulting in a new geometry generation. Repeating these steps until a desired fitness value is achieved determines the final geometry of the most successful individual. The geometry generation algorithm describes the first and maybe most influencing step of the genetic algorithm. Determining the freedom degree of the geometry generation algorithm implies setting up a definition that allows enough freedom for the algorithm to utilize as much of the solution space as possible but at the same time ensure that the solutions are fully functioning and don’t escape the desired solution space.

For the chosen example the outer shape was modified with each iteration since the purpose was to ensure more flexibility in the form design of skyscrapers and to exploit as many options as possible. One of the major optimisation criteria is minimizing wind loads on the facade of the tower so a more dynamical, organic base geometry, resulting out of lofted curves was chosen. The geometry generation definition includes a various number of flexible parameters, such as the number of curves used, the number of control points and the type of curves used. These are randomly set in the geometry definition so that one generation of individuals includes a set number of completely different geometries (Figure 1).

The second step includes the desired analysis and determination of the fitness value of each individual. This procedure also represents a critical point
since contradictory criteria cannot be 100% fulfilled but need to be weighted as to how much one criterion shall be fulfilled in comparison to the other ones. Except for choosing and implementing appropriate criteria, weighting these represents another step that strongly influences the outcome of the GA and lies in the hands of the designer. Each specific task requires different analysis criteria and weighting and moreover a strong interdependency between the analysis criteria and the geometry generation process. In the presented case the chosen criteria include a wind load analysis, wind power analysis, solar analysis, area and volume analysis and a number of excluding absolute criteria, such as minimal radii in the facade and the gravitational centre for a basic structural functioning of the building (Figures 2 and 3). The individual weighting of the criteria was performed after numerous tests according to the chosen purpose, not only to create a building as efficient as possible but also focusing on wind loads and wind power in order to achieve as little wind loads as possible on a facade pane but also to use the generated wind power with specifically located wind turbines. This is one example of clearly contradicting criteria in which less wind loads lead to a more stable structure but more wind power results in more energy win. Weighting these criteria against each other could only be achieved after a number of tests in order to understand the algorithms behaviour. In the end a minimal fitness value for wind pressure and suction was chosen to be mandatory so that the wind power generation was weighted less than the load analysis. Still using a parametric definition wind turbines were located in the areas with most wind power so that high energy efficiency could be achieved.

For the recombination and regeneration of new individual generations a stochastic selection method was chosen, such methods being acknowledged and used for an optimal reach of a solution and con-

Figure 1
Geometry generation process.
Figure 2
Fitness values of the fittest Individual.

Figure 3
Resulting tower geometry.
stant increment of the generations’ fitness values (Pinsky and Karlin 2011) (Figure 4).

The presented methods have proven to be effective for such a complex task as a high-rise project, but there were still numerous difficulties encountered along the way. One of the greatest challenges resulted to be the black box character and randomness of the written algorithm. While you can track the development of the algorithm and its success or failure, there are no means of intervening throughout the running time or even logically following the process of the genetic algorithm. It proved to be a rather random process in which you could expect that the fitness value of each individual will increase but without any means of fast tracing why or how much it increases. Many tests had to be run in order to manually try out the variable parameters, such as the weighting of the fitness criteria or the geometry generation since it couldn’t be easily understood how one value influences the algorithm. While this is part of the power of computational means, of generating designs that cannot be intuitively traced down, it remains a time consuming factor to set up all variable parameters so that a successful result will be achieved. It is a run and result process in which the designer has no capability of interacting with the computational tool he created, it is meanly a tool that needs to run through from start to end and can only then be evaluated through its result. Furthermore, a genetic algorithm needs a lot of adjustments in order to simply provide a result that constantly increases its individuals’ fitness and does not converge to a not satisfactory early result. One major point is that while the presented genetic algorithm showed effective results, it is still a linear process which follows a direction (form generation – analysis – improvement) that happens on one hierarchic level and is furthermore based on creating a very high number of random variants that are then compared and analysed. The wish for further research was to break the linearity of the algorithm and create a process in which different hierarchical levels could interact and lead to a result without the need of many variants but slowly adapting to the given requirements.

**Agent-Based Process**

Based on the knowledge of the evolutionary algorithm developed in the precedent project, a more general and flexible tool was searched in this second approach. A number of critical points discovered while using optimisation algorithms were defined as crucial and created the basis for the second approach. The exact purpose was to create a more flexible tool, with a computational core that could be extended and adapted to a given task through the addition of adaptation criteria and through changing the input constraints. As a major point the wish to destroy the linearity of such an approach and create a process that allows input parameters from various hierarchical levels and the communication between all subsystems of a general system, served as the starting point for this design method. While the evolutionary algorithm allows numerous criteria to be included and considered, it has a clear differentiation between the generating parameters, in this case the geometry generation, and the optimi-
zation criteria. The generation is the one adapting to all criteria so the information flow is unidirectional and does not allow other parameters to adapt to the requirements of the generation method. The purpose of the agent-based tool is to allow this flow of information from all input parameters into all directions and create a communication between all participating subsystems, even located in different hierarchical levels.

The chosen task is more general – also in order to exemplarily represent the possibilities of such a method – and is intended to create a roof like gridshell structure over a given fictional site. The structure is divided into three representative subsystems that are intended to show different hierarchical levels of the general structure and their interdependencies intended to allow a continuous communication between these subsystems (Figure 5).

The first chosen subsystem is the global geometry, representing the freeform surface connected to the predefined support locations. It defines the global shape and appearance of the final built result and is not intended to be only a result of all other requirements and subsystems but set and adapt according to its own constrains and requirements, such as the smoothness of the surface resulting from the angles of the different panels or the structural stability of the global force, curvature and height.

The second subsystem represents the panelling elements, such as triangles or quads through which the global geometry is realized. These have flexible parameters such as shape, or planarity. These requirements set by the designer are meant to inform the other systems while the covering panel itself shall change dimensions and orientation or location according to the requirements coming from the other systems.

The third subsystem describes a shading panel, meant to be representative for any type of facade panel reaching from a simple planar glass pane to a complex shading element. This panel again defines a set of requirements such as planarity, dimensions or orientation. As mentioned before, these systems are simply exemplary and do not cover all complexity of a gridshell structure, but are meant to show the possibilities of such a process.

For the presented example a smooth surface connecting three support locations with certain spa-
tial limitations, triangular covering panels and a simple shading component were chosen to be implemented. The agent-based system was selected after an extensive research for its capabilities of abstracting and simplifying complex behaviour into simple basic rules. An agent is defined as the smallest part of the system (the division panel) and fed with numerous rules representing all criteria of the participating subsystems. These criteria were all translated into geometric behaviour so that the agent constantly reacts and adapts to the set of requirements enabling a constant increment of these criteria.

Chosen criteria include the smoothness of the surface, dimensions of the beams, the number of beams coming together at a knot, geometric structural behaviour, static behaviour and lighting conditions. The criteria were distributed to represent a specific subsystem or be external criteria in order to show the interdependency and connections of all systems. For example, lighting conditions influence as well the shading pane that changes in size and orientation as the global geometry that is created through the individual triangular panels that focus on achieving an orientation as parallel as possible to the light source. Similarly the static behaviour influences the global geometry that tries to achieve as much double curvature as possible and through this adaptation it defines the position and dimensions of the panels (Parascho et al., 2012) (Figure 6).

One of the high advantages of this method is the flexibility of the tool and possibility to adapt it to a given task. Since the desire is to create a general tool that may be changed and fed with numerous inputs and criteria, the agent-based system was extremely efficient in allowing such fast adaptations. Each behavioural rule may be added at any time during the process and may influence all defined systems. It has also proven to be very powerful since new criteria can be implemented and tested very fast and the development can be traced simply by watching the agents perform (Figure 7).

Still a number of points have proven to be difficult when implementing such a system. The first question arising is how to abstract such a complex model as a swarm system into a working algorithm for a design purpose. It is of extreme importance how the singular agent is defined, what part of the global system it represents and how flexible it is. Defining the basic agent has the highest effect on the output, since too little flexibility may not ensure any result at all and too much will result in extreme solutions that may not be functioning as built objects.
CONCLUSION / COMPARISON

When comparing the two methods the most important fact is to differentiate between the task types that each algorithm can be addressed with. Both algorithms proved to be functioning systems for generating architectural design, but they were focusing on two distinct points. While the genetic algorithm is extremely good in handling a great number of criteria by using the high computing capacities of the computer, the agent-based model is developed to work with less information but create constant connections between this information. The agent-based model’s greatest strength is abstracting any type of criteria of any hierarchical system into one equally hierarchized level at which all parts can exchange information. It does not work through numerous variants, created with a random factor, as the genetic algorithm does, but intends to constantly change and adapt in order to improve the characteristics defined in the behaviour of the agent.

One big difference between the presented methods is the option provided by the agent-based model of interacting with the system. The black-box character of the genetic algorithm is broken as the designer can constantly follow the process of the agent-based tool. Future research will focus on the interaction with this system making use of the strength of agent-based systems to react and adapt to any exterior influence at any time. The genetic algorithm is rather a model that strongly depends on the definition of the input parameters and offers one final solution to these options. For tasks where the focus lies on the optimization process and where certain criteria need to be fulfilled as strongly as possible, evolutionary algorithms and their capacity of working with a high number of variants lead to satisfying results. On the other hands, tasks that require more adaption and fast changes in the input would rather benefit from the agent-based tool.

While both systems led to successful results, the main difficulty encountered during the processes was the correct definition of the input parameters. Whereas working through a complex solution space opens up a lot more possibilities than traditional intuitive design methods, this freedom of covering all possibilities is strongly limited by the definition of each constraint, optimization criterion or behaviour definition. Most time and energy flows into defining each parameter influencing the final output and its importance for the global tool. It is often a preconception that making use of the complexity of an architectural object through computational meth-
ods automatically opens up an unlimited space of solutions and options. The main issue which further research will address is how to define each component of such a tool as to receive the optimum balance between the freedom degree and the limitations that allow it to be a functional built object.

ACKNOWLEDGEMENTS
The implementation of the agent-based model would not have been possible without the help of Ehsan Baharlou who strongly influenced the programming of the system, ICD University of Stuttgart and the tool connecting Rhino to Sofistik developed by Riccardo La Magna, ITKE University of Stuttgart.

REFERENCES
Holland J H 1998, Emergence. From chaos to Order, Oxford University.