An early design tool for sustainable steel and steel composite structures under the use of a Genetic Algorithm

Fabian Ritter
Lehrstuhl für Computergestützte Modellierung und Simulation
Li Huang
Lehrstuhl für Metallbau
Fakultät für Bauingenieur- und Vermessungswesen
Technische Universität München
Fabian.Ritter@tum.de

Abstract: In this paper, the need for a tool for early design stages is discussed. A tool to satisfy this need, the Sustainable Office Designer (SOD) is presented. The functionality and implementation of this tool is discussed in detail. Afterwards, the output by the tool is presented. At the end, a small example under the use of the tool is shown and compared with a calculation without the use of the tool. Finally, a conclusion and outlook for further development is given.

1 Introduction

In the field of Building Information Modeling, it is becoming increasingly important to support the design team of a building project in the early design stages by providing information about the impact of decisions they make and the possibility to compare them. While there are many fully developed software tools for later design stages, the early design of the form and functionality of buildings and their structural design is supported insufficiently. But this is essential, because conceptual changes in this project phase have a major influence on later costs of the construction, facility management and deconstruction of the building.

“Until now, concept design has been a largely mental exercise of generating various spatial concepts and assessing them intuitively, based on the designer’s knowledge and accumulated expertise. Reliance on such expertise is perhaps one reason why architectural success has traditionally come only to those with decades of experience who are able to bring to bear the wisdom required to assess and select design concepts worthy of being fully developed.” (Eastman 2009).

The reason why most design decisions are made intuitively, without actual performance prediction with respect to a variety of performance aspects, such as lighting, energy, comfort, etc., is that most performance simulation tools have steep learning curves (Reichard et al. 2005). Therefore we have decided to develop a software tool which compares different typical steel and steel composite constructions for office and administrative buildings and to evaluate them regarding sustainability, flexibility, costs and energy consumption. With this, designers can be aware of the estimated performance of their creations and different structure variants generated out of them. Therefore, the effect of the modifications can be understood better. The procedure is divided into stages, i.e. architecture-structure design iterations.

2 Architectural aspects for the workflow of the tool

To develop a tool like this, it is in a first step important to understand the workflow of the future user. It has to fit in his workflow as well as support the known way of decision making without making constraints to his creativity. It has to be easy to use and to receive.

In a further step, it has to support the architect with common design knowledge. This is implemented in the tool on the one hand with rules from national standards, like minimum room heights, and on the other hand with design rules defined by architects supporting us (in name: Frank Lang and Jo Eisele from TU Darmstadt). This design rules include decisions about the construction height for a defined flexibility in the building and consumptions about how to categorize the different parts of a building in different office types and split the geometry in different bars for a better optimization (see Eisele 2011).

3 Implementation in SketchUp

In a first step a Graphical User Interface (GUI) is needed that helps the designer to draw his ideas in a simple way for the first iteration. To this end, a tool based on Google SketchUp is implemented named Sustainable Office Designer (SOD). SketchUp fulfills the criteria described above. It is an intuitive usable program which everybody can use, because it is freeware. Furthermore, it is already a common tool used by architects.

On the other hand it is a handsome tool for developing own functionalities. It is based on the programming language Ruby, which is a powerful object oriented language. With Ruby it is not only possible to create new Plug-Ins for SketchUp but also enhancing the implemented classes directly or as derived subclasses.

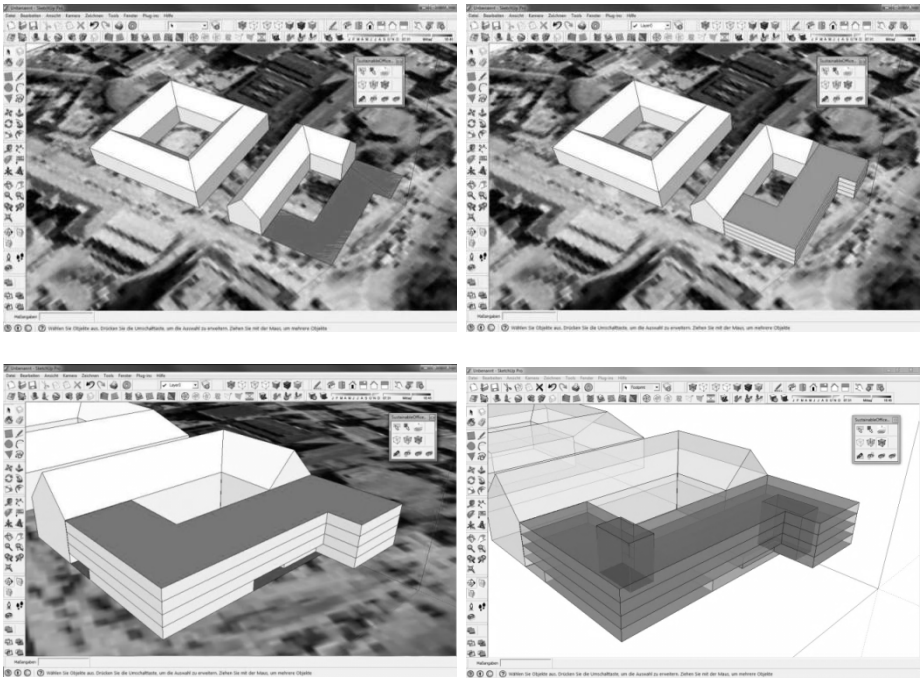

Figure 1: The workflow within the tool: The architect can draw a footprint and define the project requirements in a first step (upper left). In a second step the tool will do a first calculation and view the resulting geometry (upper right). In further steps, the architect can modify the volume with passages and entrances (lower left) and vertical circulation zones (lower right).

This is necessary, because SketchUp only provides geometric modeling. There are no features which help to make decisions based on engineering knowledge. It is not even possible to define a face as a slab or wall. But this information is needed to start a structural optimization of a building.

With this, the values needed for the optimization can be created by the input in the tool and the implemented design rules given by the architects in the project. This input serves as a basis for the structural optimization tool.

4 Programming the optimization tool

In the system a Genetic Algorithm (GA) is used to obtain an optimized solution automatically by searching for the best fitting structural system based on pre-defined parametric structures and design rules. With this, the different variants can be generated and compared efficiently. The result depends on the fitness function previously defined by the user. With the help of this fitness function it can be specified how the criteria costs, energy consumption, sustainability and flexibility are considered in the evaluation. A usable balance between these different criteria has to be found, because real life structures cannot fulfill all the best at once.

4.1 Genetic Algorithms

A Genetic Algorithm (GA) is based on the principles of inheritance and therefore belongs to the evolutionary algorithms (Eiben et al. 2007). The idea behind this is, to code certain characteristics for different options as *gen* and to store the information there. Subsequently, the different *chromosomes*, i.e. strings of *genes*, are created and compared to each other. The selection is done by a previously defined terminating function, also called *fitness-function* in accordance to the evolutionary theory. Further modifications to the selected *individuals* can be done by so called *crossings* or *mutations*. From these newly created *individuals*, again a selection is done. This happens until a defined termination criterion is fulfilled (see Mensinger et al. 2012).

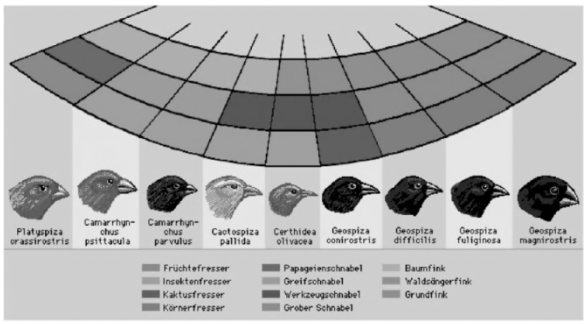

Figure 2: The basic idea behind genetic algorithms: Based on different characteristics (genes) the best fitting combination for an environment are searched. Here as an example: Darwinfinches in different forms and therefore adopted to different conditions.

Solving optimization problems normally involves discontinuous objective functions because of using simulation tools or engineering design rules. Thus, for such problems stochastic optimization algorithms are often chosen. Among many different kinds of stochastic optimization algorithms, the benefit from the GA and the reason GA is chosen, is that it is easy to implement, the optimized individuals are found very quickly, a bunch of optimized solutions at one time is available, and it is widely used for conceptual/early design phase (see Rafiq et al. 2003, Grierson & Khajehpour 2002 and Turrin et al. 2011).

For the implementation, an open source C++ library, GAlib, developed by Matthew Wall at MIT (Wall. 1996), is used. The genes representing the parameters are encoded as integer

numbers in an array using GA1DArrayGenome from the GAlib library. For the evaluation of the fitness value, a penalty term indicating the violation of design constrains was added, e.g. deflection constraints and stress constraints, while it is weighted large enough to make sure infeasible solutions die out. The design constraints are defined according to Eurocode 4 (2004) and other engineering design rules.

5 Connecting the two tools

To connect the office structure generator (OSG) programed in C++ to the SketchUp plugin, SOD programmed in ruby, SWIG is used, which generates a wrapper for ruby to use a C++ library as a ruby extension. With the help of SWIG, the office structure generator can be compiled into a .dll library that can be loaded in a ruby script as an extension. At this stage of implementation, the information of the architectural design model is passed from the SOD to the OSG as a set of parameters, e.g. the value for the dimensions of the footprint, room height, number of stories, etc. Thus, the OSG can be called within the SOD to get the optimized structure automatically generated for the architect. Since the version of ruby used in SketchUp API is 1.8.x, which doesn't support native multithreading and this leads to a problem that calling OSG freezes the main GUI, a new thread in the OSG in the C++ side to run the optimization is created. This approach solves the half of the problem that helps us avoiding frozen the GUI.

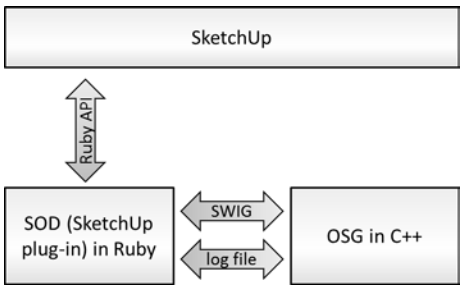

Figure 3: Structure of the Tool.

The other half is to obtain the result, i.e. the generated structure design, from the ruby side in the SOD. It is not easy to directly return the model back to the SOD, since it is in another thread created at the C++ side. So it was decided to output the result from the OSG to a log file and let the SOD check in each 0.1 second interval this file and read the parameters until the optimization finishes.

6 Output by the SOD

Another important aspect to develop an intuitive tool which supports the design process is to visualize the results from the optimization in a clear and easy understandable way. To provide this, it was decided not to show the structural model in all its details. The architect can choose a volumetric view, where he can easily understand the appearance of the building. The other view he can chose shows the structural grid and the construction heights of the slabs, to understand the structure without distracting from the overall design process. The structural grid is needed to do a proper design of the spaces inside the building but to do so the exact structure is not necessary and may only be confusing.

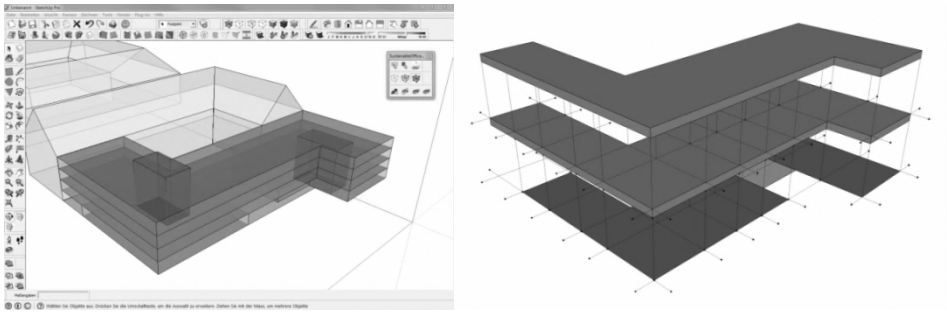

Figure 4: Visualization of the geometry as volumetric model (left) and as structural grid-model (right).

To support decision-making, it is necessary not only to provide a means for performance prediction but for performance evaluation as well. This means, comparison among alternatives is required (Papamichael et al. 1999). So another important feature of a tool for early design stages is to make the impact of the decisions made visible. Therefore, the architect can open a wind rose to see the impacts of his design and even compare it to other geometries or construction types (see Figure 5). The wind rose can be used for a "question and answer" concept as well, because it displays the new calculated values after each change of the geometry, so the architect can see the impact of his changes immediately.

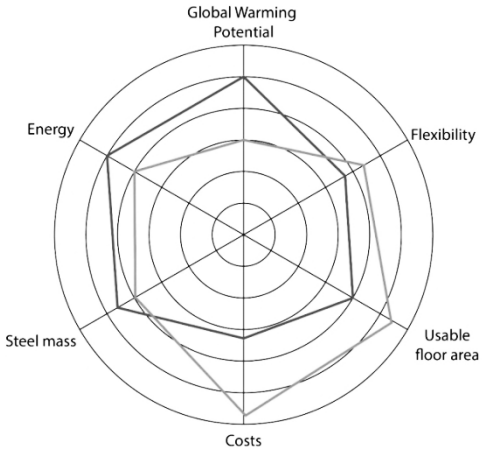

Figure 5: Visualization of the calculated values from the design tool to view the impact of the design or to compare two different designs or construction types.

The information to calculate this output comes from the members of the P881 NASTA Project (see Mensinger et al. 2011). Here it is to mention, that the tool only provides single object optimization. That means, only one of the visualized values is optimized. The other values result from this optimization e.g. the structure is optimized in regard of the costs and then the Global Warming Potential, Flexibility, etc. are calculated for this structure, without being optimized.

7 Example

GA is a heuristic algorithm which provides good solutions, but does not guarantee to find the best ones. Figure 7 shows the parametric structural model tested. The green elements are primary beams, which in this case are in longitudinal direction, can also be placed in

transverse direction. The red ones are secondary beams. The profiles for primary and secondary beams can be chosen from 71 available profiles: IPE 140 to 600 (15 profiles), HEA 100 to 600 (19 profiles), HEB 100 to 600 (19 profiles) and HEM 100 to 550 (18 profiles). All primary beams are of one profile, while all secondary beams are of another profile. Columns are separated into two groups, columns located at the inner row and columns located at the boundary. One profile is chosen for either group. IPE's are not used for columns. Fitness function is set to minimize Carbon dioxide equivalent (CDE). The building has three stories. For every story different columns are used.

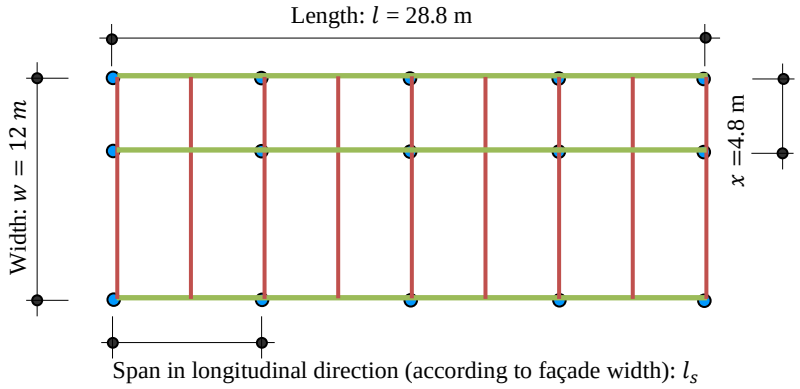

Figure 6: The parametric structural model of a rectangular floor.

We tested different combination of population size and generation size and compared the execution time and minimized fitness (see Table 1). The tests are running on Dell Precision M4500 with 8GB DDR3 and i7-840QM(1.86GHz,8MB,Quad Core,45W).

Table 1: Execution Comparison

p[1]	g	avg. time (from 5 runs)	avg. fitness (from 5 runs)
800	*1200*	*5.5 sec*	*66660*
800	*2000*	*8.9 sec*	*65460*
1200	*2400*	*18.4 sec*	*64400*
2000	*4000*	*66.2 sec*	*62764*
[1]Population Size, [2]Generation Size			

We also asked a structure engineer to propose a solution (see Figure 7) and compared it with two of the solutions obtained by the OSG (see Table 2). In Table 2 "System A/B" indicates the direction of the primary beam, while in system A the primary beam is in longitudinal direction. For the solution given by the engineer, more profiles are used for one solution.

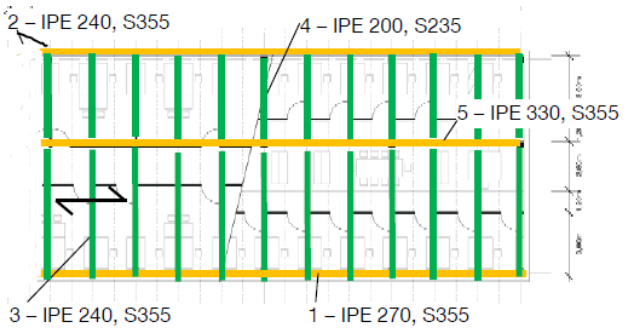

Figure 7: A structure design made by an engineer. IPE 240, IPE 270 and IPE 330 are European steel sections, while S355 and S235 are steel grades.

It has to be mentioned that beside of the comparison of the structure, there are other values not shown in the table. On the one hand it is the time to get these results. The Engineer needed half a day for this calculation. He also calculated this structure for a fixed position of columns and beams and a single system. Therefore he needs a long experience in this field to make the right decisions. Otherwise, he has to compare it with other systems, which need another half a day to be calculated. On the other hand, there is no further information about the Carbon Footprint, energy consumptions, costs, etc. of the system. To provide this, further calculations of other experts are required.

Table 2: Structure Solutions Comparison

	Geometry				**Slab**	**Secondary Beams**	**Primary Beams**	**Columns[3]**	**Fitness (CDE)**
	Position of Columns	Number of Columns	System	Number of Sec. Beams					
p[1]:800 g[2]:1200	4.8m	5	B	5	(Cofrasta, 120mm)	(IPE330, S355)	(IPE450, S355)	(HEA200,S235) (HEA120,S235) (HEA240,S460) (HEB120,S355) (HEA240,S460) (HEA160,S460)	66123
p:2000 g:40l00	4.8m	5	A	9	(Cofrasta, 120mm)	(IPE330, S355)	(IPE300, S460)	(HEB120,S460) (HEB120,S460) (HEA200,S460) (HEA140,S355) (HEA260,S460) (HEA160,S460)	61736
Engineer	4.8m	5	A	12	(Cofrasta, 140mm)	(IPE200, S235) (IPE240, S355)	(IPE240,S355) (IPE330,S355) (IPE270,S355)	(HEA140,S355) (HEA120,S355) (HEA120,S355) (HEA200,S355) (HEA160,S355) (HEA160,S355) (HEA240,S355) (HEA180,S355) (HEA180,S355)	71831

[1]Population Size, [2]Generation Size, [3]Column profiles vary for different stories and between the inner row and the outer.

8 Conclusion and Outlook

With the use of the SOD, the architect can easily design his ideas and get further information about the decisions he makes. He can rely on information from other experts in the field of engineering implemented in the tool. With the decision to use SketchUp as a basis, it is possible to provide this tool for free after completion. In further steps the possibility to do further iterations will be provided as well. The user will get the possibility to refine the

constraints and define column-free spaces, for example. Based on these new constraints, the program generates new suggestions. With each iterative step, different variants of the structure can be created and compared afterwards. On the side of the GA, the single criteria optimization will be expanded, so that the optimization addresses more than one object. So multi criteria optimization will be implemented. In the future, other systems e.g. wooden or concrete structures can easily be implemented in the tool. In this project, only steel structures are considered.

9 References

British Standards Institution. (2004). Eurocode 4: design of composite steel and concrete structures. London, BSI.

Eastman, C.M. (2009). Automated Assessment of early concept designs. In: *Architectural Design. Special Issue: Closing the Gap*., 79(. 2), S. 52.57.

Eiben, A. E. and Smith, J. E. (2007). *Introduction to Evolutionary Computing*. (2.Aufl.). Berlin-Heidelberg: Springer Verlag.

Eisele, J. (2011). *Nachhaltige Gebäudetypologien im Hochbau, Nachhaltig Planen, Bauen und Betreiben - Chancen für den Stahl(leicht)bau*. Berlin: Veranstaltung des NASTA Forschungsverbundes.

Grierson, D. E. and Khajehpour, S. (2002). *Method for conceptual design applied to office buildings. J. Comput. Civ. Eng.* 16(2), pp. 83-103.

Mensinger, M., Baudach, T., Breit, M., Eisele, J., Feldmann, M., Franz, C., Hogger, H., Kokot, K., Lang, F., Lingnau, V., Pyschny, D., Stroetmann, R. und Zink, K.J. (2011). Nachhaltige Bürogebäude mit Stahl. In: *Stahlbau*., 80, S. 740–749.

Mensinger, M., Huang, L., Zhang, P. und Hogger, H. (2012). *Optimization of sustainable office buildings in steel using Genetic Algorithms.* Oslo: Nordic Steel Construction Conference 2012.

Papamichael, K., Chauvet, H., LaPorta, J. and Dandridge, R. (1999). Product modeling for computer-aided decision-making. In: *Automation in Construction*,. 8. Elsevier Science B.V.

Rafiq, M. Y., Mathews, J. D. and Bullock, G. N. (2003). *Conceptual building design - Evolutionary approach. J. Comput. Civ. Eng.* 17(3), pp. 150-158.

Reichard, G., Papamichael, K. (2005). Decision-making through performance simulation and code compliance from the early schematic phases of building design. In: *Automation in Construction*. 14. Elsevier B.V.

Turrin, M., von Buelow, P. and Stouffs, R. (2011). *Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. Adv. Eng. Inf.* 25(4), pp. 656-675.

Wall, M. (1996). *GAlib: A C++ library of genetic algorithm components.* Mechanical Engineering Department, Massachusetts Institute of Technology.