Building Simulation 2003

Eighth International IBPSA Conference
Eindhoven, Netherlands
August 11-14, 2003

PERFORMANCE-BASED DESIGN EVOLUTION: THE USE OF GENETIC ALGORITHMS AND CFD

Ali M. Malkawi[1], Ravi S. Srinivasan[1], Yun Kyu Yi[1] and Ruchi Choudhary[2]
[1]Department of Architecture, GSFA,
University of Pennsylvania, Philadelphia, USA
[2]Taubman College of Architecture and Urban Planning,
University of Michigan, Ann Arbor, USA

ABSTRACT

This paper describes a performance-based evolution model using GA as the evolution algorithm and CFD as the evaluation mechanism. The advantages of such an evolutionary performance-based design approach is that diverse instances of the state space can be investigated in relation to specific goal requirements that will enhance the possibility of discovering a variety of potential solutions. The model allows the user to explore and visualize the design evolution and its form generation in an attempt to stimulate the designer creativity that might contribute to their output.

The process uses an iterative approach that allows design to be evaluated using CFD analysis automatically to maximize several thermal and ventilation criteria. Design change will then be performed, re-meshed and displayed based on the evolutionary algorithms. The process allows the user to experience the morphing of the design based on its performance and continues until the designer has the opportunity to visualize the evolution of the final set of design alternatives.

INTRODUCTION

Using thermal and ventilation performance criteria as a mean to generate creative design alternatives requires an in-depth knowledge of the problem and considerable expertise. Each design change made to a room geometry, or room boundaries requires the user to re-model, re-mesh and subsequently, re-compute the airflow. Trade-off between different design changes generally remains obscured because of the complexity of the model. As a result, the design space is difficult to explore systematically and solutions can be overlooked. Evolutionary techniques coupled with visualization can offer one solution to such a problem.

Evolutionary algorithms have traditionally been used to solve optimization problems. In addition, they can be used as a design aid. The evolutionary approach is a generate and test approach which corresponds well to the procedures for design synthesis and evaluation in the design process. In an evolutionary-based generative process, design representations are specified as a set of parameters, and a corresponding set of constraints. Generative design describes a broad class of design where the design instances are created automatically from a high-level specification.

With the advent of new technologies in the field of design evolution, the designer's role shifts from that of a creator of individual instances of a style to that of a meta-designer or creator of an entire style of family (Soddu, 1997). Such generative designs assist in the rapid generation of computable design descriptions, design models or design representations (Chien et.al., 2002). In addition, these design evolutions can be used as an aid in stimulating the designer creativity (Todd et.al.,1992). The advantage of such an evolutionary approach is the creation of diverse sections of the state space that increases the possibility of discovering a variety of potential solutions (Rosenman et.al., 1994).

Recently, several research projects were conducted that took advantage of performance-based design approach (Lam, 2002; Michalek et.al., 2002; Bouchlaghem, 2000). Although these studies illustrated variety of approaches that target optimizing design variables for variety of evaluation criteria, they did not attempt to capture all possible design variable evolutions which provide larger search space for the designers to interact with.

This paper builds on an earlier work that integrated computational fluid dynamics and genetic algorithms (Choudhary et. al., 2002). It presents a performance-based design evolution model that allows efficient exploration of design alternatives using a four-layer approach: design evolution, performance evaluation, morph visualization, and design evaluation.

The design evolution uses a GA module that generates the shape of the design instances. For each design change, a CFD analysis is automatically performed to evaluate the thermal and ventilation efficiency, and the results are fed back into the GA module. This process runs recursively till the best designs are returned.

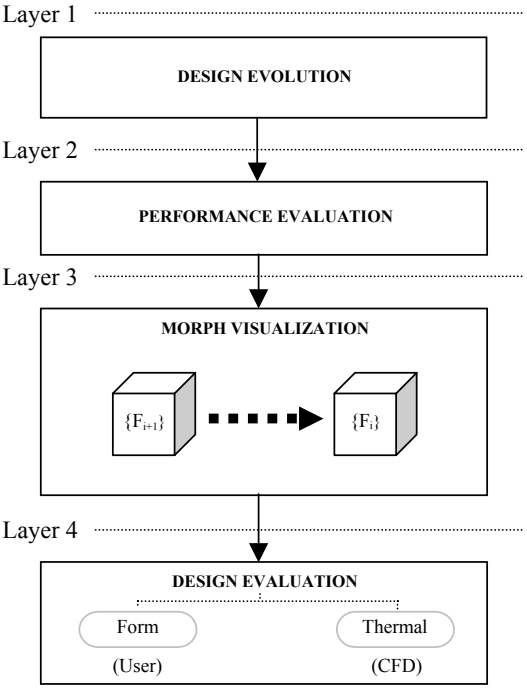
To capture the system's solutions and in-between instances as the solution is evolving, the system is integrated with a morphing module. This module

allows the diverse instances of designs to be visualized along with their performance in order to enhance the user's potential of design discovery and provide aid within the design process.

During morphing, users can intervene to stop the process and select an instance based on its form. This design instance can then be evaluated for its performance using an integrated CFD engine. Temperature and velocity contours are automatically generated for users to visualize and check for performance criteria. This evolutionary design approach creates discrete instances of designs that satisfy the performance targets by providing larger search space for designers to interact with.

THE PERFORMANCE-BASED DESIGN EVOLUTION STRUCTURE

The performance-based design evolution framework consists of a four-layer structure, figure 1. It uses GA as the evolution algorithm and CFD as the evaluation mechanism. The model runs recursively till the best designs are identified. This eliminates the need for the user to specify, design and arrange a display each time an output is required. All possible design evolutions can be visualized and their performance can be predicted.



F_i = Forms generated based on geometry targets by GA.

Figure 1: Four-layer Performance-based Design Evolution model.

The model is implemented to allow three-dimensional orthogonal geometries to be presented. The space is defined by length, width and height and contains a window, a door, one supply duct and an extract terminal. Supply velocity is computed on the basis of room volume, area of the supply ducts and air change per hour required for the room. In all, the problem constitutes of twenty-one design variables: three discrete and eighteen continuous, figure 2. Each variable is associated with explicit upper and lower limits and the interval over which it could be incremented.

Design Evolution

In the Design Evolution layer, design representations are specified as a set of variables and parameters, figure 2. The variable and parameter-sets form the genetic elements of the design, and defines the shape of the design instances. This formulation allows for the generation of various forms (F_i) through the recursive generation of its components.

	Variables	Parameter(s)
Performance Targets		
Target temperature		T_T
Supply temperature	S_{Tmax}, S_{Tmin}	S_T
Supply velocity	S_{Vmax}, S_{Vmin}	S_V
Geometry Targets		
Room Length	R_{Lmax}, R_{Lmin}	R_L
Room Width	R_{Wmax}, R_{Wmin}	R_W
Room Height	R_{Hmax}, R_{Hmin}	R_H
Window location		W_{Wx}, where $x^* = n,e,w,s,f,r$
Window left distance	W_{Lmax}, W_{Lmin}	W_L
Window right distance	W_{Rmax}, W_{Rmin}	W_R
Window floor distance	W_{Fmax}, W_{Fmin}	W_F
Window ceiling distance	W_{Cmax}, W_{Cmin}	W_C
Supply duct location		S_{Wx}, where $x^* = n,e,w,s,f,r$
Supply X distance	S_{Xmax}, S_{Xmin}	S_X
Supply Y distance	S_{Ymax}, S_{Ymin}	S_Y
Supply length	S_{Lmax}, S_{Lmin}	S_L
Supply height	S_{Hmax}, S_{Hmin}	S_H
Return duct location		R_{Wx}, where $x^* = n,e,w,s,f,r$
Return X distance	R_{Xmax}, R_{Xmin}	R_X
Return Y distance	R_{Ymax}, R_{Ymin}	R_Y
Return length	R_{Emax}, R_{Emin}	R_E
Return height	R_{Imax}, R_{Imin}	R_I

[*n = north, e = east, w = west, s = south, f = floor, r = roof]

Figure 2: Variables of Layer 1

Since this is a mixed continuous – discrete variable problem, GA is used to explore the design space. Apart from explicit upper and lower bounds, other constraints have been included to ensure the feasibility of the design before it is evaluated. These

include constraints on the window size and constraints on the location of supply and extract terminals to prevent overlapping. The genetic algorithm used is GALib, a C++ genetic algorithm library. Since GALib does not handle constraints explicitly, the window size and overlapping constraints are modeled as penalty functions, and included in the design evaluation stage as the first design evaluation step (Choudhary et. al, 2002).

Performance Evaluation

The Performance Evaluation layer uses CFD to perform the thermal analysis. Gambit v2.0.4 is used for mesh generation followed by Fluent v6.0, figure 3. If there is a physical configuration error or inappropriate simulation setup, the CFD does not converge, prompting the design to be penalized. This is accomplished by providing a penalty value that is returned back to layer 1. This value is used to reject the evolution of such future instances and to ensure the generation of different designs. If CFD analysis converges to a solution, the values are passed to layer 3 for post-processing of thermal and ventilation performance data and further morphings.

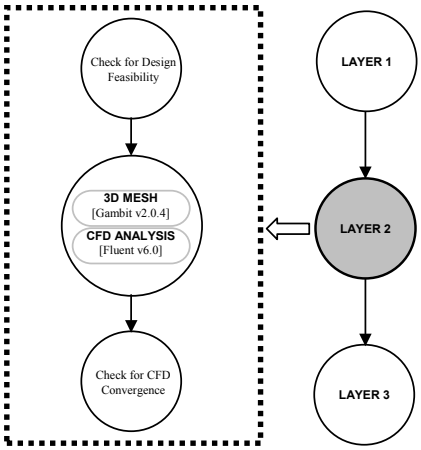


Figure 3: Layer2 – Performance Evaluation.

Morph Visualization

The Morph Visualization layer aids in the design evolution process by visualizing the intermediate designs, which may, in normal circumstances be difficult to locate, figure 4. Morphing is an integral part of design evolution since it forms a smooth transition between pairs of consecutive keyframe models. The morphing module generates design instances between two consecutive forms analyzed by the CFD (between F_{i+1} and F_i) to evolve new forms (F_{mi}) through "m" time intervals:

F_{i+1} ----------> F_i [through „m" time intervals]

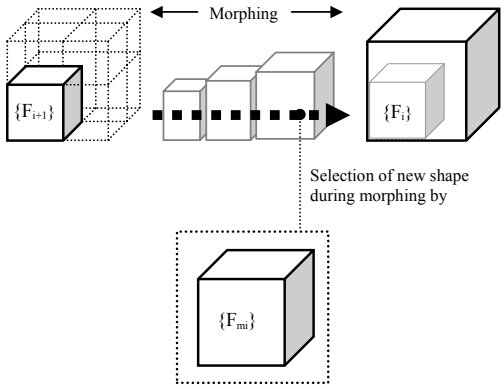


Figure 4: Layer3 – Morph Visualization.

The forms used by the module represent three-dimensional orthogonal spaces. The morphing process ensures that the form instances selected are always unique in their geometry. The new forms evolve with changes in room [length, width, height], window (left distance, right distance, floor distance, ceiling distance), figure 5, and supply inlet duct and return outlet duct (x distance, y distance, length, height), figure 6.

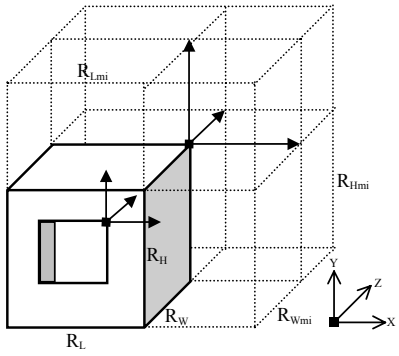


Figure 5: Design flexibility exhibited for room and window forms.

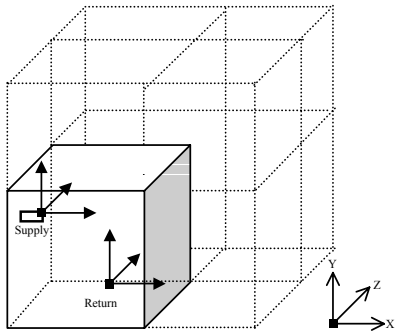


Figure 6: Design flexibility exhibited for supply and return ducts.

For every run of the GA algorithm, one form is generated. Therefore, morphing does not exist during the first cycle. For subsequent cycles, morphing process runs continuously with a fixed time interval. Since the morphing is a direct linear transformation between two consecutive forms, any in-between shape can be identified as:

$$\{[F_{i+1} \pm F_i] / m\} \times \{\text{selected time-frame as a ratio of 'm'}\}$$

where " m" is the time interval in milliseconds.

Therefore, any in-between user-selected shape (denoted as F_{mi}) during the morphing process is again a set of geometric design variables:

F_{mi} represents $\{R_{Lmi}, R_{Wmi}, R_{Hmi}, W_{Lmi}, W_{Rmi}, W_{Fmi}, W_{Cmi}, S_{Xmi}, S_{Ymi}, S_{Lmi}, S_{Hmi}, R_{Xmi}, R_{Ymi}, R_{Lmi}, R_{Hmi}\}$

where " i" = 1, 2, 3, ... n

Design Evaluation

The Design Evaluation layer allows user intervention to take place. The intervention begins by the user selecting an instance from the evolving shape. The morphed shape can then be evaluated for its performance, figure 7. Performance can be evaluated by invoking the CFD engine. A mesh is automatically generated and sent with the corresponding boundary conditions for the CFD engine to be executed. The CFD outcome of temperature and velocity contours can then automatically be generated and channeled back to the Graphic User Interface (GUI).

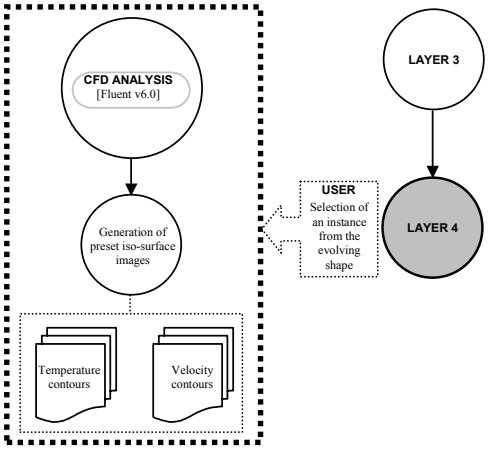


Figure 7: Layer 4 – Design Evaluation.

SYSTEM INTERFACE

The system interface was developed using Java API. The interface uses function calls to interact with the GA module, Gambit v2.0.4 and Fluent v6.0. Several GUI mechanisms were built to allow better control of the system. These GUIs aid in data-entry, modification and visualization in different modes. Data entry and modifications are used to manipulate form and its properties. Visualization aids the user to explore the design evolution and its form generation.

The interface provides access to both morphing and performance evaluation. The interface allows the user to select their mode of visualizing the forms, either as wireframe or shaded models, figure 8. Once the CFD converges to a solution, the morph GUI is displayed. This GUI is composed of three sections. The two sections at either corner of the GUI represent the two form instances (F_{i+1} and F_i) and their performances, while displaying the continuous morphing process in the center (F_{mi}), figure 9. During the morph process, the user can zoom, rotate and scale the form instances dynamically for better visual perception.

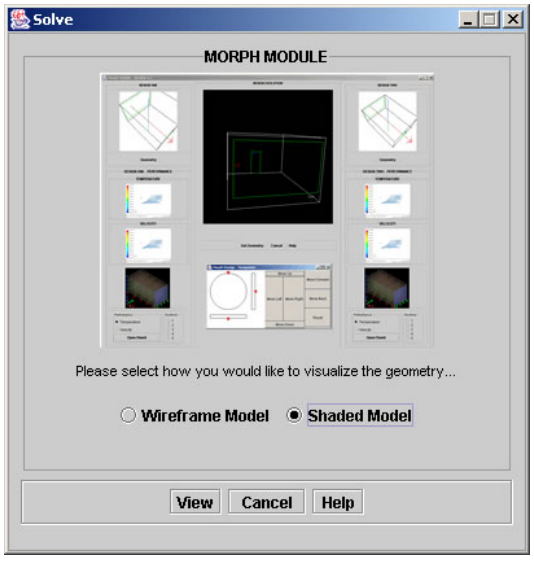


Figure 8: Model-mode selection GUI.

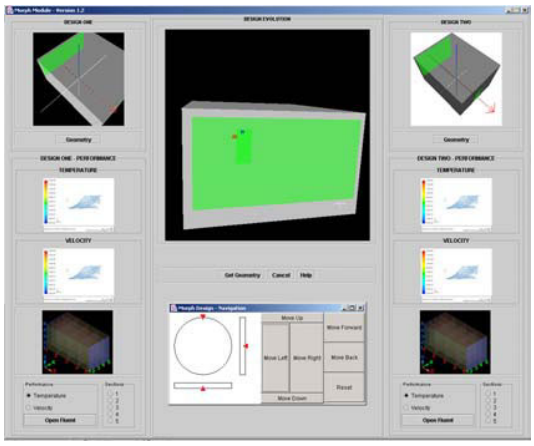


Figure 9: Morph-process GUI.

The performance of the form instances can be visualized directly by selecting the iso-surface image, figure 10. Five equi-distant iso-surfaces along each

axes are provided. Depending on the user selection, the corresponding CFD outcome of temperature and velocity contours are displayed enabling the user to visualize the performance of the instance being investigated, figure 11.

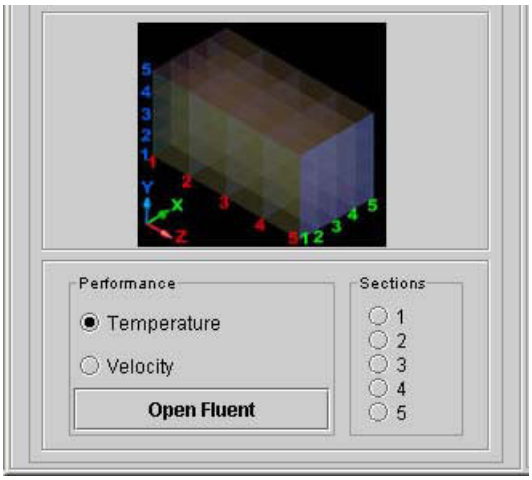


Figure 10: Preset iso-surface image selector.

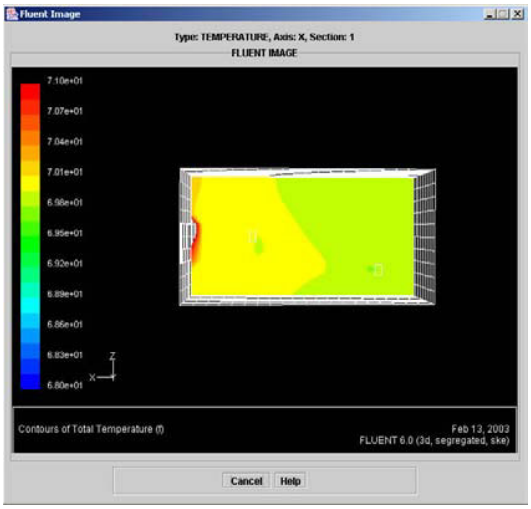


Figure 11: A typical display of temperature contour.

CONCLUSIONS

The framework described demonstrates the potential of exploring and visualizing the design evolution and its form generation based on a set of performance targets. Furthermore, this framework (a) captures all possible design evolutions, (b) provides a larger search space for the designers to interact with, and (c) enables the user to visualize the performance of any design instance selected during morphing.

By integrating an optimization module that creates discrete instances of design with a morphing module, that captures these consecutive instances and morphs the instances in between, the system was able to provide an example of continuous evolution of optimization.

The morph component promises solution(s), which may still not be the "optimum" but takes into consideration the designer preferences. It allows the user to intervene in the search process and select designs based on the designer preferences. These design instances represent points in the design space that are close to the GA selected instances, but may not be the output of the algorithm because their performance is similar to the one selected by the GA. In this manner, the user has active control over selecting and evaluating designs during the design generation process.

One of the primary objective of performance-based design is to optimize the performance targets specified by the user – thermal and ventilation criteria, in this case. The effectiveness of a particular design to satisfy the performance targets, measures the value of that design. Thus, if the design satisfies both requirements effectively, it could be referred to as a good solution. Although these discrete design instances are considered to be good solutions, it is difficult to ascertain that any intermediate instance during morphing would be a good solution that maintains the performance targets unless it is analyzed by the performance module. Current work includes enabling these intermediate instances to be also evaluated automatically. In addition, the design representation is being modified to include non-orthogonal geometries and other thermal boundaries.

These modifications will test system scalability and will allow more complicated forms to be generated. In addition, it will allow the possibilities of evolving a wide range of design performance scenarios such as tilt of glass, configuration of atriums, etc. that can be utilized during the next user evaluation phase.

ACKNOWLEDGEMENTS

This work was funded by a grant from the University of Pennsylvania Research Foundation.

REFERENCES

Bouchlaghem, N. 2000. Optimizing the design of building envelopes for thermal performance, Automation in Construction, Vol. 10, pp 101-112.

Chien, Sheng-Fen, Flemming, U. 2002. Design space navigation in generative design systems, Automation in Construction Vol. 11, pp 11-22.

Choudhary, R. and Malkawi, A. 2002. Integration of CFD and Genetic Algorithms, 8th International Conference on Air Distribution in Rooms, Copenhagen, Denmark.

Lam, K.P, Mahdavi, A., Gupta, S., Wong, N.H., Brahme, R., Kang, Z. 2002. Integrated and distributed conputational support for building performance evaluation, Advances in Engineering Software Vol.3, pp 199-206.

Michalek. J, Choudhary. R, Papalambros. P. 2002. Architectural Layout Design Optimization, Engineering Optimization Vol. 34(5), pp 461-484.

Rosenman, M.A. and Geor, J.S. 1994. The what, the how and the why in design, Applied Artificial Intelligence, 8(2): 199-218.

Soddu, C. 1997. Argenic Design, European Academy of Design Contextual Design / Design in Context Conference, Stockholm, Sweden.

Todd, S. and Latham, W. 1992. Evolutionary Art and Computers, Academic Press, London.