Technical Report

Applying Genetic Algorithm to Geometry Design Optimization

Improving Design by Emulating Natural Evolution

By

Zheng Yi Wu, Ph.D, Research Director
Pradeep Katta, Research Intern
Applied Research Group, Bentley Systems, Incorporated
27 Siemon Company Dr, Suite 200W
Watertown, CT 06795, USA

Note: This report summarizes the research project undertaken by Pradip Katta under supervision of Zheng Yi Wu at Bentley, Systems, Incorporated, Watertown, CT office during the summer of 2009. The work can be cited as follows.

Executive Summary

Geometry modeling has become an increasingly powerful approach for architecture and building design. It is an effective approach especially when a geometry model is constructed by making use of the associative parameters, so-called algorithm-based parametric design, which enables designers to easily change the desired geometry parameters and thus fine tune the design. This report elaborates a new approach to leverage the use of parametric model for optimizing a geometry design by using genetic algorithm (GA), a search method based on the principles of natural evolution and genetic reproduction.

An integrated design tool has been implemented for the optimization of parametric geometry design. Based on evolutionary search algorithm and geometry modeling tool such as Generative Component, a design can be represented by encoding design variables onto a binary string or genotype, design alternatives are evolved by mimicking crossover, mutation and natural selection principle of Darwin's survival-of-fittest. The tool allows user to select any combination of parametric graph variables and geometry parameters to optimize the design. A design solution is evaluated by using the fitness score that can be any user-defined geometry attribute. One fitness score is assigned to each of new designs. The fitter the solutions are, the more likely the solutions are selected to reproduce next generation of design solutions. Thus solutions are optimized generation after generation via emulating natural evolution. The integrated tool has been tested on a simple case of solid design and also on a real use case of sport stadium design.
Table of Contents

Executive Summary .................................................................................................................. 2

Table of Figures ...................................................................................................................... 4

1 Introduction .......................................................................................................................... 5

2 Literature Review ................................................................................................................. 6

3 Design Evolution .................................................................................................................. 8

3.1 Solution Framework ......................................................................................................... 8

3.2 Solution Methodology ..................................................................................................... 11

3.2.1 Parametric Model ........................................................................................................ 11

3.2.2 Interactive Genetic Algorithm ..................................................................................... 11

3.3 Integrated Implementation .............................................................................................. 16

3.3.1 Graphical User Interface ............................................................................................ 17

3.3.2 Engine Module ........................................................................................................... 22

3.3.3 Design Creation Module ............................................................................................. 23

3.3.4 Design Evaluation Module ......................................................................................... 23

3.4 How it works ..................................................................................................................... 24

3.4.1 Creating Manual Design Run .................................................................................... 24

3.4.2 Creating Optimization Design Run ............................................................................ 29

4 Applications ....................................................................................................................... 33

4.1 Solid Design ..................................................................................................................... 33

4.1.1 Design without Constraint ......................................................................................... 34

4.1.2 Design with Volume Constraint ................................................................................ 36

4.2 Sport Stadium Design ..................................................................................................... 38

4.2.1 Initial Design and Challenges .................................................................................... 39

4.2.2 Minimum Unique Triangles ....................................................................................... 40

4.2.3 Quadrilateral Spacing ............................................................................................... 42

5 Conclusions ........................................................................................................................ 44

5.1 Advantages ....................................................................................................................... 44

5.2 Limitations ....................................................................................................................... 45

5.3 Future Work ..................................................................................................................... 46

6 References ............................................................................................................................ 46
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Framework for Interactive Evolution Design</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Implementation Overview</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Integration with DE Engine</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Implementation of DE Application</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>New Menu Button</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>Run Menu Button</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>Design Evaluation Tab</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Engine Parameters</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Result Table</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>DE GUI with an active manual run node</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>Initializing DE Application</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>DE GUI with persisted data</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>GUI with welcome screen</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>New Manual Design Features and Properties</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>Manual Design with new Values</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>New Manual Solution Node</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>Manual solution node details</td>
<td>29</td>
</tr>
<tr>
<td>18</td>
<td>New Optimized Design Run Features and Properties</td>
<td>30</td>
</tr>
<tr>
<td>19</td>
<td>Optimal Design Run with input parameter values</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>Optimal Design Run engine parameters</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>Initial Generation Optimal Solutions</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>First Generation Optimal Solutions</td>
<td>33</td>
</tr>
<tr>
<td>23</td>
<td>Generation Number Vs Average and Best fitness score for case 1</td>
<td>36</td>
</tr>
<tr>
<td>24</td>
<td>Generation Number Vs Average and Best fitness score for case 2</td>
<td>38</td>
</tr>
<tr>
<td>25</td>
<td>Conceptual design of sport stadium by BIAD</td>
<td>38</td>
</tr>
<tr>
<td>26</td>
<td>generated grid for creating polygon mesh</td>
<td>39</td>
</tr>
<tr>
<td>27</td>
<td>Generated quadrilateral polygon mesh</td>
<td>39</td>
</tr>
<tr>
<td>28</td>
<td>Area statistics of the optimized triangles</td>
<td>41</td>
</tr>
<tr>
<td>29</td>
<td>Optimized triangles with horizontal spacing</td>
<td>42</td>
</tr>
<tr>
<td>30</td>
<td>Quadrilateral spacing for reconstructed triangles</td>
<td>42</td>
</tr>
<tr>
<td>31</td>
<td>Reconstructed triangles with quadrilateral spacing</td>
<td>44</td>
</tr>
</tbody>
</table>
1 Introduction

Computer-aided drafting (CAD) and analysis tools have been developed for architectural geometry design as well as for building system modeling. Equipped with intuitive, user-friendly and integrated CAD and analysis modeling tools, architects and engineers are able to accomplish the design that is more complex than ever in more efficient manner than before. Today CAD tools are well developed for representing geometry components, especially parametric associative approach provides designers a tool to not only generate the design but also maintain the geometric relationships among the graphical features. For instance Bentley’s Generative Component (GC) is such a tool that allows designers to undertake parametric geometry modeling. Such a CAD modeling tool offers advantages in that users can explicitly define the geometry relationships that are intuitively presented and even programmatically manipulated.

However, the parametric associative model provides limited function for effectively exploring design alternatives, users have to manually tune the design by adjusting the parameters (e.g. predefined graph variables) to arrive with a final design. The design quality still very much depends on how many design alternatives have been explored to arrive with the final design. There is constantly lacking of the function in today’s CAD tools to facilitate effective exploration of design space.

Therefore, a need has arisen to develop computing method and tool that can automatically create design alternatives and hopefully evolve the design according to pre-defined design criteria. The aim of this research project is to develop a prototype framework and tool for interactive evolution of architecture, building and geometry design. The approach is based on interactive genetic algorithm (GA), a computation method of simulated evolution. The GA is used to generate and evolve the design alternatives, each of which is represented by using a GC transaction file. A design is evaluated by a fitness score. The design with the greater evaluation score is more likely selected by GA than the one
Applying Genetic Algorithm to Geometry Design Optimization: *Improving Design by Emulating Natural Evolution*

with less evaluation score to generate next generation of designs. Generation after generation, the design is expected to be optimized by the simulated evolution process.

2 Literature Review

A review on previous research has been conducted, primarily focusing on the works presented at Smart Geometry forum over last two years, the work completed by academic research groups including the emergent design group joined by MIT architecture department and artificial intelligence lab, the design integration research by TU Delft in collaboration with Arup the Netherlands, architecture geometry design projects undertaken by TU Vienna Austria as well as the latest international conference on Advances in Architectural Geometry that was held in Sept. 13-16 2008 in Vienna Austria.

Martin Hemberg (2001) from MIT emergent design group has done some early research in applying evolutionary computing method to evolutionary design. A prototype program Genr8 has been developed as plug-in to Maya Autodesk 3D modeling platform. The research work employed L-System a rule production method to automatically grow a design from the base geometry elements (such as curves, surfaces and meshes etc.) that are defined by user. A new design is generated by a certain grammar representing the rules. Each possible design is evaluated by weighted design criteria including the size, smoothness, symmetry, subdivisions, soft boundary, undulation etc. Some preliminary results have been achieved with Genr8 that is only working with outdated version of Maya. There are more related research projects undertaken by MIT emergent design group, including Weaver (Peter Testa 2001) for exploring industrial braiding and weaving; MoSS (Testa 1999) for generatively growing surfaces and GERMZ (Testa 2000) using genetic programming to interactively generate 3-D form. Most of them were implemented in C/C++ as a plug-in to Alias/Wavefront studio. A relative recent research on design exploration has been undertaken by Axel Kilian (2006), who, in particular, experimented with using evolutionary computing methods (Genetic algorithm, genetic programming etc.) for optimizing developable surfaces from a predefined mesh. Kilian and the group at TU
Austria (2008) have further extended the approach for developable surfaces with curved creases, which represents some most promising outcome but intriguing challenges remains for generalizing the method as geometry modeling tool.

Coenders and the group at TU Delft have done some significant research work in interfacing Bentley’s GC with structure analysis software. A set of design tools have been prototyped for interfacing GC with Arup’s GSA (Oasys 2006) and Tekla Structure (Tekla 2006). The integration has set an example to interface GC with other structure analysis software tools. The seamless integration will facilitate the fast and accurate computation of finite element method and computational fluid dynamic analysis required for performance-based design. Although this work does not directly enable iterative optimization of a design, but it serves as important component for fast evaluating a GC design within the framework of optimization design modeling.

In parallel to other research groups’ work in evolutionary design, Hiroaki Nishino and his colleagues at Oita University in Japan have done good research in developing interactive design model for 3D graphics design. Nishino’s research since 2000 has been focused on applying interactive evolutionary computation to 3D digital model. The basic idea is to use evolutionary computing method to automatically generate a set of design alternatives (3D graphics), each design will be evaluated or scored (in different categories e.g. “very good”, “good” and “not good”) by human designer. The evaluation scores of the designs will be passed to evolutionary computing method to evolve next generation of designs. The iteration can proceed as long as desired to obtain one design and/or a number of final designs. In the latest published research by Nishino (2008), an immune algorithm (IA) simulating the mechanism (antibody formation and self-regulation) of human immune system was employed for evolving the design alternatives.

The integrated method by Nishino does not only allow the optimization of the graphics shape but also the physical appearance. The graphics is represented by using Java 3D graphics library and optimized by changing clay modeling options (twisting and tapping) as well as altering control mesh nodes while the physical appearance is optimized
by rendering operations including light sources, surface materials and background colors. An intuitive prototype digital modeling tool has been developed and large amount of 3D graphics have been interactively evolved by using the tool.

Based on the literature review in the area of evolutionary design for digital graphics, architectural and geometry design, it is clearly indicated that there is strong interest in developing intelligent design tool to enable effective exploration of design space. In particular, as indicated by Carlo Sequin (2008) as follows.

“CAD tools are most helpful today in the final phases of design, where a lot of the validation depends on much detailed, tedious computation, which humans gladly offload to machines. Today's CAD tools are probably the least helpful at the very beginning of the design process in the initial, creative phase of conceptual design.”

Therefore, in order to develop the CAD function for assisting with initial creative design, it is necessary and imperative to undertake the research in developing a general framework and tool that permit the exploration of design space.

3 Design Evolution

Currently architects and engineers use associative and parametric modeling system such as Generative Components (GC) for exploring alternate building forms without manually building the detailed design model for each scenario. GC helps in designing several designs by manually tuning the design parameters. It is a trial and error process and usually more time consuming. Thus only a limited number of design alternatives are explored.

In order to overcome the above mentioned problem we developed the design evolution (DE) tool. DE is a design optimization tool that provides user with a number of optimal solutions (design alternatives of the original design) by optimizing the desired design parameters.

3.1 Solution Framework
The framework for Design Evolution (DE) is designed to work along with GC. As shown in figure 1, it consists of several integral parts, including (1) DE Graphic User Interface (GUI) that allows users to start DE, set up and manage the design evolution runs; (2) Initial Design module that allows user to select and load up an initial design; (3) Define Design Variables module that allows users to choose the desired parameters for optimization and set parameter ranges (minimum and maximum values for each parameter, as well as the corresponding step); (4) Parameter Evolution by genetic algorithm optimization engine that performs the artificial evolution of the selected parameters with specified steps and within the specified ranges; (5) Design Generation that creates multiple designs with the each set of design parameter values and (6) Design Evaluation module where new designs are scored by human evaluation and passed back into GA optimization engine. The modules of GA optimization engine, creating new design and human-based evaluation forge an interactive evolution process of Design Evolution.
In the simple words the implementation can be described using as in Figure 2. It optimizes the selected design variables represented in GC design file or the transaction file. The tool is developed as a feature in the GC environment although it can be developed as a standalone application if deemed necessary in future. Therefore, the current scope of this application is limited to GC transaction file, but the method is generic and applicable to any parametric geometry model.
In brief, DE Application implemented are working interactively with GC environment and Design Evolution Engine during the process of generating new designs. The basic functionality of this application can be outlined as follows:

1. Get the input design to be optimized.
2. Provide user with a Graphical User Interface (GUI), which enable user to choose the design parameters and specify their range.
3. Optimize the design parameters using Design Evolution engine, and create new design files.
4. Evaluate the designs and continue to next generation.
5. Repeat steps 5 and 6 until satisfactory or near-optimal solutions are obtained.

The functions outlined above describe the basic steps of DE implementation. The detailed solution methodology for the integrated application is given as follows.

3.2 Solution Methodology

The main task of DE prototyped tool is to automatically generate design alternatives and optimize the design by applying genetic algorithm. Therefore, the core solution methodology is to integrate a parametric model with design evolution engine.

3.2.1 Parametric Model

Generative Components (GC) is the parametric geometry model that is used to load the initial design to be optimized and presents the alternative designs generated by Design Evolution. GC application programming interfaces (API) are extensively used to access the design parameters that are displayed in DE GUI where the user can intuitively select design parameters and set up optimization runs.

3.2.2 Interactive Genetic Algorithm

Genetic algorithm is the type of search and optimization method that is based on the principles of natural evolution and genetic reproduction. It represents a solution by using a string or so-called chromosome. An initial population of strings is randomly generated, each of the solutions is
evaluated with a fitness value or score to measure its merit or strength to survive competition for being selected as parents to reproduce new solutions (offspring). A new population of solutions is generated by emulating genetic reproduction operations e.g. crossover and mutation. Generation after generation, the solutions are expected to evolve towards the satisfactory or near-optimal solution. Such a genetic algorithm engine is the core part of the solution method as shown as in Figure 3 in the dashed rectangle. It is developed to work interactively with user via three components including:

1. Engine solver
2. Engine APIs
3. Engine data

![Integration with DE Engine](image)

**Figure 3 Integration with DE Engine**

**Genetic Algorithm Solver**

There are many different types of genetic algorithms. In this research project, the fast messy genetic algorithm (fmGA) method, originally developed by Goldberg et al. (1993) has been applied to geometry design evolution. Over last decade, the fmGA solver has been improved and successfully applied to a number of Bentley products including Darwin Calibrator (Wu et al. 2002), Darwin Designer (Wu et al. 2002) and Darwin Scheduler (Wu 2004), as well as research projects in water quality model calibration (Wu 2006), sensor placement optimization (Wu 2009), leakage hotspot detection and extended period model calibration (Wu 2009), parallel pump scheduling (Wu and Zhu 2009) and structure damage detection (Wu and Xu 2009). The
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

method has proven to be effective and robust for solving a wide range of engineering optimization problems.

As shown in Figure 3, DE interacts with fmGA solver via the engine APIs, together with the input and output data that are handled through three text files for the simplicity of the implementation. The engine solver is enhanced to work in such an interactive way that the engine computation progress can be fully controlled by user through GUI. The engine can be initiated, paused and stopped as desired. All the controls are provided by a set of application programming interfaces.

**Engine Programming Interfaces**

Application programming interfaces are developed for employing fmGA solver to solve the problem of geometry design optimization. APIs are a set of methods that provide the functionality to control the engine and enables DE application to:

1. Initiate engine. The fmGA solver will be initiated with user specified input data.
2. Continue to next generation. A new population of solution will be created while fmGA will proceed with next generation.
3. Pause and Stop engine. fmGA computation will be either paused or terminated.

**Data Handling**

DE engine uses three kinds of data including design input, solution output and fitness score. Each of the data type is described as follows.

*Design Input*

The input data include design variables and the parameters used by the engine for optimization purpose. For each design variable to be optimized, user should specify its range. The range includes minimum value, maximum value and the increment value for that parameter. A set of default optional GA parameters are provided for GA search while users are also allowed to change those parameters. Input data should be created by GUI and stored in a simple text file of
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

a predetermined format, which can be easily transformed into a data schema. The sample input dataset is given as follows.

```
[DESIGNVARIABLES]
; Name         Min  Max     Increment
point01-XTranslation  1.0  2.0   0.5
point01-YTranslation  0.1  0.8   0.1
point02-XTranslation  4.0  8.0   0.5
point02-YTranslation  4.0  6.0   0.5
point02-ZTranslation  2.0  5.0   1.0
point03-XTranslation  1.0  2.0   0.5
point03-YTranslation  6.0  10.0  0.5

[OPTIONS]
Evaluation type  Human
Number of top solutions  3

[GAPARAMETERS]
Maximum era Number  6
Era generation number  20
Population size  30
Cut probability  0.017
Splice probability  0.60
Mutation probability  0.015
Random Seed  0.50
```

In the above file, design variables are those design parameters that the user is interested to optimize. For each of the design parameters the user needs to specify the range as discussed. DE GUI also enables the user to specify the initial population size and the other engine parameters, which are given as input to the engine.

Solution Output

For each generation of fmGA optimization, fmGA engine outputs a population of design solution into a text file. For the above input file, the sample solution file is given as follows:

```
SOLUTION  0
point01-XTranslation  0.50
point01-YTranslation  0.50
point02-XTranslation  7.50
point02-YTranslation  0.50
point02-ZTranslation  2.00
point03-XTranslation  0.50
```
Each of the solutions is used to generate a design or a GC transaction file, which will be evaluated with a fitness score to quantify the merit of the design solution according to the predefined criteria.

**Fitness Score**

Once initial population is created, the fitness of each solution should be calculated and written into a score file. The engine uses the fitness values to proceed with creating next generation of solutions. This process should be repeated till the satisfactory or near-optimal solutions are obtained. In our DE application, a design evaluation file (GC script file) is created to evaluate the design files created after each generation and write the corresponding fitness values into the engine score file. The sample score file is as shown below.

<table>
<thead>
<tr>
<th>s1</th>
<th>15.07</th>
</tr>
</thead>
<tbody>
<tr>
<td>s2</td>
<td>4.10</td>
</tr>
<tr>
<td>s3</td>
<td>8.43</td>
</tr>
<tr>
<td>s4</td>
<td>13.49</td>
</tr>
<tr>
<td>s5</td>
<td>13.49</td>
</tr>
<tr>
<td>s6</td>
<td>4.54</td>
</tr>
<tr>
<td>s7</td>
<td>3.12</td>
</tr>
<tr>
<td>s8</td>
<td>10.4</td>
</tr>
<tr>
<td>s9</td>
<td>12.54</td>
</tr>
<tr>
<td>s10</td>
<td>7.33</td>
</tr>
<tr>
<td>s11</td>
<td>2.33</td>
</tr>
</tbody>
</table>

The solution method is implemented and integrated with GUI to perform the design optimization. More details are given as follows for the prototype implementation.
3.3 Integrated Implementation

In order to perform various functionality of our application, the tool is implemented with four integral modules namely:

1. DE GUI.
2. Engine Module.
3. Design creation module.
4. Evaluation module.

The complete implementation of our application is as shown in the Figure 4. The arrows represent the direction of data flow. For instance, DE GUI creates a dataset as input for fmGA engine while fmGA engine produces design solutions and output to DE GUI.

Figure 4 Implementation of DE Application
Our DE application is a simple and user friendly application. Role of each module and their functioning is explained in the next section of this chapter. All other modules except the fmGA engine module use the GC API’s. Engine module uses the engine API’s for interacting with the design evolution engine.

### 3.3.1 Graphical User Interface

The DE graphical user interface (GUI) acts as a front end for our application. Our GUI mainly consists of two panels side by side.

**Left Panel**

Left panel in DE GUI consists of

1. Menu strip on the top.
2. Tree view of the solutions.

Menu strip on the top consists of four menu buttons including

a) New menu button
b) Delete button
c) Rename button
d) Run button

Delete and rename buttons are used to delete or rename a particular node in the tree view. Deleting a parent node delete all of its child nodes. While a child node is deleted, that particular node is removed from the tree view without affecting the other nodes.

As shown in the Figure 5, the new menu button has two buttons in it. Our DE application gives the user an option to create a manual designs as well as optimal designs.

Manual designs are created by generating a new design file with the parameters specified by the user without optimizing them. In the optimal design run, several optimal solutions are created by optimizing the design parameters.
Applying Genetic Algorithm to Geometry Design Optimization: *Improving Design by Emulating Natural Evolution*

Figure 5 New Menu Button

Figure 6 Run Menu Button
Run menu button as shown in the Figure 6 consists of four buttons. New manual design button helps to generate one manual design with the parameters specified by the user.

Initial Optimized run button helps to start the DE engine. Next Generation Optimized Design Run button makes the engine to move to the next generation. User can also stop the engine by clicking on the stop design evolution button.

**Right Panel**

Right panel consists of several data tables where user can view and enter data. Based on the design run the corresponding data table is displayed where user can enter the input parameters. Once the design run is executed the results are shown in the result table.

As shown in the Figure 7, the geometric variables tab in the right panel displays the available design parameters on the top. Once the user clicks on the design parameter, corresponding feature properties are displayed. In Figure 7, the property values of de-eval01 feature are displayed and user can specify the range which includes minimum, maximum and increment values for each of the feature properties to be optimized.
As shown in the figure above, the optimized parameters that correspond to the optimal solution -0 are displayed in the design evaluation tab and its corresponding design score is displayed in the design score field on the top. When the optimal solution is selected, the corresponding optimized values and their design score are displayed in this tab.

![Figure 8 Engine Parameters](image)

Options tab as shown in the Figure 8 displays the engine parameters that user can change. These parameters play an important role in the optimization. Figure 9 shows the result table that displays scores of each optimal solution created. Rows define the generation number and columns define the optimal solution number in the corresponding generation. Average score value, best score value in a generation are also displayed.
Tree View

Tree view displays nodes in a organized manner. Once a new manual or new optimal design run is selected, the corresponding node is added to the tree view. After each generation, the corresponding generation node with its generation number is added. The optimal solutions created during that generation are added as its child nodes.

The GUI for a manual design run is as shown in the Figure 10. One can compare this figure to Figure 5 to observe that difference that as this is a manual design run, the user need not define the range for the parameters. He can just specify the new value. Once the new manual design run button is clicked, new design file with the user defined value is created.
Engine module plays an important role interacting with the design evolution engine using the engine API’s. This is mainly responsible for controlling the engine in the following ways:

1. Start the engine.
2. Take the engine to the next generation.
3. Stop the engine.

When the Initial Optimized Design Run button shown in the Figure 6 is clicked, the engine module starts the DE engine and passes the input file. The input file is created based on the range on design parameters specified by the user.
Before the engine continue to the next generation, engine module checks that the new design files are created and are evaluated.

### 3.3.3 Design Creation Module

This module is responsible for creating optimal design files after each generation. This module relies on the GC API’s for creating new designs.

For the initial general optimized run, the DE engine creates an output file which is discussed in the earlier in this chapter. For the next generations, this output file is updated. This output file is accessed by this module and creates a new design file for each optimal solution in the output file. This new design file is created by using GC API’s and replacing the old values of the parameters with the new ones.

### 3.3.4 Design Evaluation Module

Design Evaluation module takes care of evaluating the newly created designed and feed their fitness score to the engine. The fitness value can be any positive real number. These fitness values play an important role in producing the optimal solutions. In general using different evaluation methods for evaluating same design result in different optimal solution. Hence design evaluation methods play an important role in determining the optimal solutions. For example if the fitness score chosen to be a number between 1 and 10, 1 represents a very bad design and 10 represents a very good design. The fitness score helps the DE to discriminate between good population and bad population.

In order to make a design evaluation an automatic process, design evaluation module of our DE application generates a design evaluation script file. This is nothing but a GC script file. This file iterates through the solutions created after each generation. The score or fitness value associated with each solution is written in a score file (a text file).

The fitness value of each solution can be computed using different evaluation methods. For example in one of our test cases which we presented in the next chapter the ratio of area to volume of the solid is used a design evaluation method.
Some of the other design evaluation methods of interest are:

1. Structure Analysis.
2. Energy Analysis.

### 3.4 How it works

As discussed DE application can be used to create both manual designs and optimal designs. In this section the working of our DE application is explained in a step wise manner.

#### 3.4.1 Creating Manual Design Run

**Step 1:** Either open or create an initial design in the GC environment. The design must be reasonably well, representing the basic functions that the design needs to provide.

**Step 2:** Start the Design Evolution tool, which is implemented as a feature in the GC environment as shown in the following Figure 11.
In order to access our DE application, it should be loaded into the GC environment as an assembly. In order to do that go to Tools->Manage Loaded Assemblies->Select Precompiled Assembly File and load our (DEFeature.dll) which is provided to you.

Once the assembly is loaded into the GC environment, the DE Application is visible with the name DEFeature as shown in the Figure 12. You can pass any of the features in the input design to initialize the application. By default the name of this feature will be defeature01.

Once the Ok button is clicked our DE GUI is displayed. Before loading the GUI DE application checks if data corresponding to this design file is persisted. If so it will display the previous run nodes and all the other persisted data as shown in the Figure 12. This implies that user had worked on this design file and ran one optimize design run and one manual design run. The user has option to view all the interested data. The main advantage of this data persistence is that it prevents data loss.

Figure 12 DE GUI with persisted data
If there is no persisted data, you see our DE GUI with a welcome screen as shown in Figure 13. And the new menu button is activated enabling the user to choose either manual design run or optimal design run. As this section demonstrate the creation of a manual design, click on the new manual design run button.

![Figure 13 GUI with welcome screen](image)

**Step 3:** Select the features and set a new value to their properties that need to be varied to create a new design.

As shown in the Figure 14, the available design features and their feature type are displayed in the top table. Once a feature is selected their corresponding properties are displayed in the bottom table. The current value of the property is also displayed in the table. The user needs to give a new value to the property which he is interested to change. For example change solid type to 2 and solid size parameter to 15. Figure 15 shows the user inputs.
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

Figure 14 New Manual Design Features and Properties

Figure 15 Manual Design with new Values
**Step 4:** Click the new manual design run button.

After providing the new values and new manual design run is clicked, a new manual design is created with the user defined parameter values in the same folder as the input design file with name “manual design-0.gct”. A child node is added to the tree view which corresponds to the manual design file created. If this node is selected, the optimal parameters in the new design file are displayed as shown in Figure 17. The current value and new value are displayed.

![Image of New Manual Solution Node](image)

*Figure 16 New Manual Solution Node*
The above four steps demonstrate how to create a new manual design using our Design Evolution application.

### 3.4.2 Creating Optimization Design Run

The steps 1 and 2 remain same for the manual design run and optimal design run.

**Step 1:** Either open or create an initial design in the GC environment. The design must be reasonably well, representing the basic functions that the design needs to provide.

**Step 2:** Start the Design Evolution tool, which is implemented as a feature in the GC environment as shown in Figure 11.

**Step 3:** Select the features and set a new value to their properties that need to be varied to create a new design.
If you compare the Figure 18 with Figure 14, you need to enter the range of the parameters to be specified as opposed to new value in manual design run. User need to specify the minimum, maximum and increment values for each of the parameter that need to be optimized.

**Figure 18 New Optimized Design Run Features and Properties**

**Figure 19 Optimal Design Run with input parameter values**
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

Figure 20 Optimal Design Run engine parameters

Set the engine parameters which includes the number of design alternatives to be generated for each iteration/generation and other related optimization settings as shown in Figure 20.

**Step 4:** Click on the initial Optimized Design Run button.

Initial Generation of the population is generated by the DE engine. As shown in the Figure 21 the initial set of optimal solutions are created and are added as nodes to the tree view. Clicking on the optimal solution node display the corresponding optimal solution parameters in the design evaluation in the same way as the manual design run. DE application creates an input file based on the user input as discussed earlier. DE file uses the engine solution file to create new optimal design files.
Step 5: Evaluate the Designs and write the fitness score into the engine score file.

As discussed in this chapter, the design evaluation module creates the design evaluation script file, which is in the same folder of the initial design. Open the evaluation file in a new GC environment and execute all the transactions. This ensures that all the designs are evaluated and their corresponding scores are written into the engine score file.

Note: User should update the evaluation score attribute in the evaluation script file before running it, e.g. Print ("s"+scoreFileIndex+"\n"+line01.Length) is to use the line length as optimization criteria or the evaluation score attribute.

Step 6: Click on the next generation optimized design run button.

This moves the engine to the next generation and the next set of population is generated as shown in Figure 22. The user can keep track of the best solutions by looking at the result table as shown in Figure 9. The result table is updated after every generation.
Step 7: Repeat steps 5 and 6 until the best optimal solutions are obtained.

The above steps demonstrate creation of optimal solutions using our DE application.

Our DE application implements data persistence so that user can save the work session before closing it.

4 Applications

In order to test the effectiveness of the developed DE prototype, the tool has been first tested on a simple design case as follows.

4.1 Solid Design

The test case used for the testing purpose is a solid design. The initial design is created as a GC transaction file that produces one of these solids including cube, cylinder, torus or sphere based
on the user input. The other design parameter is the size of the solid. Based on the value of the size parameter, and the type of solid specified, GC generates the corresponding solid design.

As discussed earlier, different evaluation methods result in different optimal solutions. This is because different evaluation methods result in different fitness score which in turn result in different optimal solutions generated by the DE engine. Thus fitness function used for the evaluation purpose also plays a key role in the designs generated by our DE applications.

For this test case, a solid design is evaluated by using the ratio of the solid surface area to solid volume. In general, the greater the surface, the greater the cost, thus solid surface area can be treated as a surrogate of construction cost. The volume of a solid is the space provided by a design. Therefore, the ratio of the surface area to volume represents the cost per unit design space. The solid design with the least ratio is corresponding to the minimum cost. In order to achieve the least cost design, the task is to optimize the solid design so that the minimum ratio can be reached for a solid shape and size. The design optimization is conducted for two scenarios of with and without volume constraints.

### 4.1.1 Design without Constraint

In this scenario we wanted to find a solid with the minimum ratio of surface area to volume without any volume constraint. A solid is designed by the parameters given as follows.

1. Shape type parameter, noted as \( S \in \{1, 2, 3, 4\} \), where 1 is cube, 2 is cylinder, 3 is torus and 4 is sphere.
2. Solid Size Parameter \( R \in [1, 5] \)

The objective or fitness function is defined as:

\[
F = \frac{A}{V} \tag{1}
\]

Where \( A \) is the surface area and \( V \) is the volume of the solid. The optimization design is given as:

Search for: \((S, R)\) \tag{2}
Minimize: \( F \) \hspace{1cm} (3)

The optimization design has been undertaken using DE with the results obtained as follows.

The optimized design solutions are presented in Table 1, which shows the solid type and the fitness value at the bottom of the solid associated with it. The average fitness is the average fitness value for that generation. And the best fitness is the minimal fitness value of the best solid type and size found up to the generation.

Table 1 Design solutions without volume constraint

<table>
<thead>
<tr>
<th>Generation</th>
<th>Best Fitness</th>
<th>Avg. Fitness</th>
<th>Design solutions (shape and fitness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial generation</td>
<td>0.042</td>
<td>0.677</td>
<td><img src="shapes1.png" alt="Shapes" /></td>
</tr>
<tr>
<td>Generation 10</td>
<td>0.042</td>
<td>0.319</td>
<td><img src="shapes2.png" alt="Shapes" /></td>
</tr>
<tr>
<td>Generation 20</td>
<td>0.033</td>
<td>0.073</td>
<td><img src="shapes3.png" alt="Shapes" /></td>
</tr>
<tr>
<td>Generation 30</td>
<td>0.033</td>
<td>0.054</td>
<td><img src="shapes4.png" alt="Shapes" /></td>
</tr>
<tr>
<td>Generation 40</td>
<td>0.032</td>
<td>0.052</td>
<td><img src="shapes5.png" alt="Shapes" /></td>
</tr>
<tr>
<td>Generation 45</td>
<td>0.031</td>
<td>0.049</td>
<td><img src="shapes6.png" alt="Shapes" /></td>
</tr>
</tbody>
</table>
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

Figure 23: Generation Number Vs Average and Best fitness score for case 1

Figure 23 represents a graph that is plotted between generation number and the average and best value associated with it. From the results, it is clear that our application found the sphere to be a solid with minimum fitness which can be proved in theory as well.

### 4.1.2 Design with Volume Constraint

This scenario is aimed at finding a solid which has minimal ratio of surface area to volume with the target volume specified by the user. The design parameters are the same as scenario I except a volume constraint has been posed, thus the optimization design is given as:

Search for: \((S, R)\) \hspace{1cm} (4)

Minimize: \(F\) \hspace{1cm} (5)

Subject to: \(V \leq V_t\) \hspace{1cm} (6)

In order to effectively handle the design constraint, a penalty function is constructed as

\[
F = \begin{cases} 
\left( \frac{A}{V_t} + \frac{V_t}{V} \right) & ; V - V_t \leq 0 \\
\left( \frac{A}{V_t} + \frac{V_t}{V} + \frac{(V - V_t) \times 50}{V_t} \right) & ; V - V_t > 0
\end{cases}
\hspace{1cm} (7)
\]
Where \( A \) is the surface area of the solid, \( V \) is the volume of the solid and \( V_t \) is target volume of the solid (specified by the user), for this test case we used \( V_t = 25000 \). The results are given in Table 2. The output we got is a sphere with surface area of 4117 and volume of 24838. Figure 24 illustrates the convergence rate for the design scenario with target volume constraint.

Table 2 Design solutions with volume constraint

<table>
<thead>
<tr>
<th>Generation</th>
<th>Best Fitness</th>
<th>Avg. Fitness</th>
<th>Design solutions (shape and fitness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial generation</td>
<td>2.6</td>
<td>596.2</td>
<td><img src="initial_generation_images" alt="Images of design solutions" /></td>
</tr>
<tr>
<td>Generation 10</td>
<td>1.7</td>
<td>3.1</td>
<td><img src="generation_10_images" alt="Images of design solutions" /></td>
</tr>
<tr>
<td>Generation 20</td>
<td>1.7</td>
<td>24.2</td>
<td><img src="generation_20_images" alt="Images of design solutions" /></td>
</tr>
<tr>
<td>Generation 30</td>
<td>1.4</td>
<td>5.2</td>
<td><img src="generation_30_images" alt="Images of design solutions" /></td>
</tr>
<tr>
<td>Generation 40</td>
<td>1.4</td>
<td>6.1</td>
<td><img src="generation_40_images" alt="Images of design solutions" /></td>
</tr>
<tr>
<td>Generation 45</td>
<td>1.4</td>
<td>2.0</td>
<td><img src="generation_45_images" alt="Images of design solutions" /></td>
</tr>
</tbody>
</table>
4.2 Sport Stadium Design

The developed DE prototype has also been tested on a sport stadium design in China. The initial design has been undertaken by Beijing Institute of Architecture Design (BIAD). Figure 25 illustrates the conceptual design of the stadium. One of the architecture design tasks is to construct the roof surface of polygon mesh.
4.2.1 Initial Design and Challenges

The initial design was done by using Bentley Generative Component (GC). A number of GC transactions are implemented to create two curves on both edges, points on the curves, arc segments over the curves, a grid of points along the arc segments as shown in Figure 26, and finally an initial quadrilateral polygon mesh is generated as shown in Figure 27.

Figure 26 generated grid for creating polygon mesh

Figure 27 Generated quadrilateral polygon mesh

The initial surface design of the stadium has more than 20,000 quadrilateral panels, which are different in size and shape, manufacturing each distinct panel is expensive to construct the roof surface. To reduce the cost, it is desirable to reduce the number of unique polygons. Thus the task is to optimize the mesh design such that the number of unique panels is minimized.
4.2.2 Minimum Unique Triangles

Because the initial design was constructed by generating a mesh of quadrilateral polygons, it is difficult to proceed with optimization due to that (1) the quadrilateral panels are non planar and (2) it is difficult to compare them for congruence. Therefore, the entire roof is recreated with triangles that ensure a planar panel and easy to check for congruence among the triangles.

Based on the discussion with BIAD architects, the optimization is set up with DE tool as follows.

1. Design variables
   
   (a) The number of points on the arc segment, vary from 89 to 109;
   
   (b) X and Y coordinates of two points on law curve;

2. Fitness is specified as the number of unique triangles and calculated by comparing the triangle congruence.

Using the initially constructed triangle mesh, a new optimized triangle mesh is constructed in order to minimize the number of unique triangles. The triangle reconstruction algorithm is as follows.

   Step 1. Sort triangles by all three sides;
   
   Step 2. Group by two short sides within the specified tolerance e.g. 0.05, 0.1, 0.15 and 0.2.
   
   Step 3. Find two shortest sides within the group;
   
   Step 4. Find the vertex with two shorter sides, then reconstruct the triangle by replacing two sides with two common sides as identified at Step 3;
   
   Step 5. Calculate the unique number of reconstructed triangles by comparing all the reconstructed triangles for congruence with a tolerance of 0.05 m used for side comparison.

   It is found that the tolerance used for aggregating triangles (step 2 above) is the dominant factor in minimizing the number of unique triangles. With the different grouping tolerance, the number of unique triangles is minimized as shown in Table 3. As illustrated, the greater the tolerance, the less number of unique triangles is required to reconstruct the roof surface. For the strictest case, 78% of unique triangles are reduced for the grouping tolerance of 0.05 m. For the other more relaxing tolerances, even greater reductions of unique triangles have been achieved.
Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

The statistics on the reconstructed triangles are also undertaken as presented in Figure 28 for the case of 0.1 m grouping tolerance. It shows that the most of the triangles are having area of less than 5 square meters. The reconstructed roof panel is also illustrated in Figure 29, which shows a unitary spacing along the horizontal line. This pattern of spacing is hiding away the paneling view of quadrilaterals that is originally desired by BIAD. Therefore a quadrilateral spacing is considered for constructing the triangles.

![Figure 28 Area statistics of the optimized triangles](image)

Table 3 The number of unique triangles for different tolerances for grouping

<table>
<thead>
<tr>
<th>Grouping Tolerance (m)</th>
<th>Number of Unique Triangles</th>
<th>Reduction Percentage of Unique Triangles (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>2785</td>
<td>88</td>
</tr>
<tr>
<td>0.15</td>
<td>3317</td>
<td>87</td>
</tr>
<tr>
<td>0.10</td>
<td>4001</td>
<td>84</td>
</tr>
<tr>
<td>0.05</td>
<td>5395</td>
<td>78</td>
</tr>
</tbody>
</table>

Total Number of Triangles: 24,794
Figure 29 Optimized triangles with horizontal spacing

4.2.3 Quadrilateral Spacing

As discussed in the previous sections, the roof surface is constructed with triangles. In order to achieve the quadrilateral view, a regular spacing must be left on the two shorter sides of each triangle, as shown in Figure 30, for instance, spacing must be left along the sides of ac, bc, ad and bd while ab is the shared or common side for both triangles within one quadrilateral.

Figure 30 Quadrilateral spacing for reconstructed triangles

In order to have the regular spacing and to minimize the number of unique triangles, new triangles a’b’c’ and a’b’d’, as illustrated in Figure 30, are reconstructed as follows.
Step 1. Find point $a'$ and $b'$ such that $aa'$ and $bb'$ is the desirable spacing e.g. 0.05 m;
Step 2. Find the midpoint $m$ of the side $ab$ that is shared by both triangles;
Step 3. Construct median line segments $mc$ and $md$ for the triangles;
Step 4. Find point $c'$ on line segment $mc$ and $d'$ on line segment $md$ such that the new triangles $a'b'c'$ and $a'b'd'$ can be congruent.

Due to the strict requirement for the regular spacing such that the panels can be viewed as quadrilaterals, there is not much freedom to reconstruct triangles except finding the new points of $c'$ and $d'$ along the median line segments. Following methods are considered for determining the new vertices on the median line segments.

1. Find new vertices $c'$ and $d'$ with the desired spacing along the median line segments, that is to leave the same spacing (e.g. 0.05) between the corresponding old and new vertices. It helps to reconstruct the triangles with uniform spacing.
2. From midpoint $m$ of the shared line segment $ab$, find new vertices with the shorter median line segment of two triangles. The minimum spacing of 0.05 m is left for the shorter median line segment. (unique triangles: 8688, 65% reduction)
3. From midpoint $m$ of the shared line segment $ab$, find new vertices with the shortest median line segment of a group of triangles. The following method is developed to aggregate the original triangles.
   a. Sort the triangles by the common side, this ensure two adjacent triangles within the same quadrilateral are aggregated into the same group;
   b. Group the sorted list of triangles by the common side, with tolerance of 0.05, resulted in about 400 groups;
   c. For each group, further aggregating the triangles by the median line segment with tolerance of 0.05, it creates 1652 groups.

Three methods as outlined above have implemented for reconstructing the triangles. They have ensured regular quadrilateral spacing as shown as in Figure 31. The triangles congruence is checked by comparing corresponding sides with tolerance of 0.05. The numbers of unique triangles are as shown in Table 4. A total of 64% or more has been achieved for reducing the unique triangles in shapes and sizes. However, in comparison, there are more unique triangles generated by quadrilateral spacing than by horizontal spacing.
### Table 4 The number of unique triangles for quadrilateral spacing

<table>
<thead>
<tr>
<th>Methods for find vertex on median segments</th>
<th>Number of Unique Triangles</th>
<th>Reduction Percentage of Unique Triangles (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform spacing</td>
<td>8688</td>
<td>65</td>
</tr>
<tr>
<td>Two adjacent triangles</td>
<td>8594</td>
<td>65</td>
</tr>
<tr>
<td>Triangle aggregation</td>
<td>8994</td>
<td>64</td>
</tr>
</tbody>
</table>

Total Number of Triangles: 24,794

---

**Figure 31 Reconstructed triangles with quadrilateral spacing**

### 5 Conclusions

This document is mainly intended to give an overview about project and its implementation. The results presented in the previous chapter suggest that our application produced desired outputs and the results are promising as well.

#### 5.1 Advantages
Developing interactive evolutionary design with GC is the first endeavor in undertaking the applied research projects of the integrated design modeling. It is to facilitate the process of performance-based design in order to improve the infrastructure sustainability. The outcome of the initial research is to produce a prototype that provides adequate functions to demonstrate the potential benefits of DE, which include, but may not limit to:

5.1 Capability of better exploring more design alternatives than using geometry modeling tools (e.g. GC) alone. DE engine is to generate new designs that human expert never thought about by simulated evolution.

2. Flexibility of optimizing user-selected design parameters that may include all the predefined graph variables and additional feature parameters.

3. User-guided and controlled design optimization process in which each new design will be scored by human experts and thus new designs are generated by combining the features of better-scored designs.

4. Plug-in to GC and Microstation is beneficial for user to easily integrate with existing design process to improve the design quality and efficiency.

5.2 Limitations

Provided that the tool is to be developed as planned with the multiple benefits, however, the pitfalls still exist in applying the tool, especially when the tool falls into wrong hands. The interactive design tool is to generate and optimize the design according to user-given scores, thus the quality of the design will completely guided by the evaluation. The final design is just as good as human designer’s evaluation. However, the tool is not to automatically ensure a good quality solution but help user to achieve a good solution in a cost-effective manner.
5.3 Future Work

The research project undertaken thus far can be extended to develop the methods and integration prototypes for automatic design evaluation, including:

I. Generic geometry evaluation. In order to streamline the evaluation, new GC APIs are to be developed for loading transaction file, accessing graphic attributes and playing transactions.

II. Design performance evaluation. This includes rapid analysis of energy consumption and structure responses. Develop and integrate the performance-based evaluation will greatly improve the practical applicability of the design optimization too.

Finally, the stadium design case also revealed that a design case in the scale could take much time to complete the run of geometric evaluation for one design solution. It is worthwhile exploring the research in high performance cloud computing to speed up the efficiency, which has been demonstrated in other research projects.

6 References


Applying Genetic Algorithm to Geometry Design Optimization: Improving Design by Emulating Natural Evolution

Smart Geometry Forum at http://www.smartgeometry.org


47