Creative Tower Generated by Computational Intelligence

Xiyu Liu[1], Hong Liu[2]
[1]School of Management & Economics, Shandong Normal University, Jinan, China
[2]School of Information Science & Engineering, Shandong Normal University, China
Email: *xyliu@sdnu.edu.cn*

Abstract. Based on previous works by the authors on evolutionary architecture paradigm of evolving architectural form, new research is being carried out to develop a virtual environment with system implementation model of evolving creative tower designs. The solid models are created by evolution with computational intelligence and enhanced mathematical models. Models will evolve according to computational intelligence including generic algorithms, particle swarm optimization. Mathematical models include analytical functions, parametric functions and other nonlinear functions. We also present analysis of relationship between evolution and exploration.

1. Introduction

Computational intelligence and computer modeling have been efficient ways in architecture design (18). In this area much work has been achieved in natural model, evolutionary models and revolutionary models (Frazer, 2002). In fact, computer modeling has become so important that one can hardly find any design without the help of it. And there are more and more artworks and designs that are called generative art which have widely changed the conventional idea. As the work of professor Celestino Soddu shows (http://www.celestinosoddu.com/), generative art is one of the new ideas that can get artificial objects. In this way, we can work producing three-dimensional unique and non-repeatable shapes.

Of the many computer aided design tools up to now, most of them provide hand-made utilities, and a designer has to draw every sketch to get the outline of a product (Frazer, 2002). And many generative tools are based on sketches. These limitations motivate the conceptual design methodology introduced in this paper, which is capable of using mathematical functions to get novel shapes.

The purpose of this paper is to report our new research in evolutionary towers by computational intelligence. We will introduce some relations of computing to design

– a new bridge of design and enhanced computation algorithms. This is possible because there is usually not a single optimal design for any problem, but rather designs evolve. Concerning to computing design, we will introduce (1) computing tools that is useful for creating and improving design alternatives, (2) creation of conceptual resources that is helpful in order to create design concepts, (3) linear and nonlinear algorithm for improving basic design concepts towards successful solutions. A new system is developed to implement the design process with enhanced computation techniques and complex functions. The system kernel is compatible with object-oriented technology and component reusing. An evolutionary architecture paradigm with a focus on how visionary and creative forms can be achieved is demonstrated. We will present our complex form generation and visualization system with images and rapid prototyping models that are otherwise impossible to generate by normal CAAD systems without using generative and evolutionary computation.

A new type of genetic algorithm is studied for our generative design system. We extend the classical powerful techniques from modern nonlinear analysis theory to selection and optimization of GA. These techniques include topological spaces and partial ordering. A Zorn Lemma type of iterative procedure is introduced. This attempt will partially overcome the difficulty in implementing effective automatic selection in the application of genetic algorithms.

2. Mathematical models and 3D shapes

Apart from its computation functionality, mathematics indeed reveals the beauty of nature. From symmetry to structure, from honeycomb to skyscraper, even giving a glance to the crowding cars, we will find the beauty of mathematics everywhere.

However, it is not easy to look into this beauty without the help of artist or computer. Of the early literatures, the book of Gerd Fischer (Gerd, 1986) is a successful one that introduces the elegance and beauty of mathematics. This book collected 132 images taken from real models of the most important mathematical models, including differential geometry, projective models. It is this book that changes the viewpoint of many people who always take mathematics as abstract, and who reveals the beautiful and symmetric structures that are potentially virtual models of architectures.

Nowadays, one of the most significant ways to understand a mathematical model is computer visualization. However, due to the fully nonlinear nature of many functions, it is not an easy work to develop accurate shapes for general nonlinear functions. One way to solve this problem is the finite element method. There are several approximate techniques for nonlinear functions. The simplest is planar piece. In his work, Peter J. Bentley (1996) uses primitive shapes consist of a rectangular block or cuboids with variable width, height and depth, and variable three dimensional positions to construct nonlinear objects. His blocks are intersected by a plane of variable orientation in order to approximate curved surfaces. More accurate techniques include nurbs approximation, polynomials approximation and others. It should be noted that one can hardly find the balance between a better approximation and acceptable computing time.

With the help of solid modelling libraries, we developed 3D solid model visualizations for complex functions in this project. A prototype system has been implemented based on an integration of ACIS 3D solid modelling kernel and MatLab with a C++ graphical user interface. Our basic geometrical objects for approximation are nurbs surfaced units. Our system is fully compatible with commercial CAAD tools and systems, as well as rapid prototype facilities. A large number of object-oriented components of sophisticated surfaces and envelopes based on taxonomy of generic form have been built. In particular, complex forms are classified as linear, quadratic, trigonometric function, exponential functions, root functions compounded functions, rotations, sphere and cylinder co-ordinates, implicit function. Computational mechanisms have also been developed with which these basic data structures and components can be visualized, combined or split to allow new data structures or new forms to be derived using generative techniques.

3. Theoretical structure: the concept of homotopy

What is homotopy? Intuitively, a homotopy is a constant deformation from one shape to another according to some rules. Although homotopy itself is an important concept in geometry and topology, we only borrow the idea here to describe our problem. These rules are called homotopy map. By defining various maps we can generate different homotopies. However, we should remember that there are indeed shapes that are not homotopic, that is, we can not find homotopy to transform one into another.

Fig3.1 is a curve illustrating homotopy. The left one is the original curve while transforming to the final curve, the right one. Intermediate curves show that homotopy is a continuous shaping process.

Whenever we use homotoopy to generate towers, the outline of the building is often the most attracting factor. This is the second factor just following its high and extruding illusion in a city landscape. In fact, a special tower in a city even becomes the representation of the city. In this paper, our main concern is to study generated towers with outlines determined by mathematical functions and evolution. We use a multi-coding schema to represent phenotypes, that is, the continuous schema corresponding to continuous functions, and the discrete schema corresponding to discretization of functions.

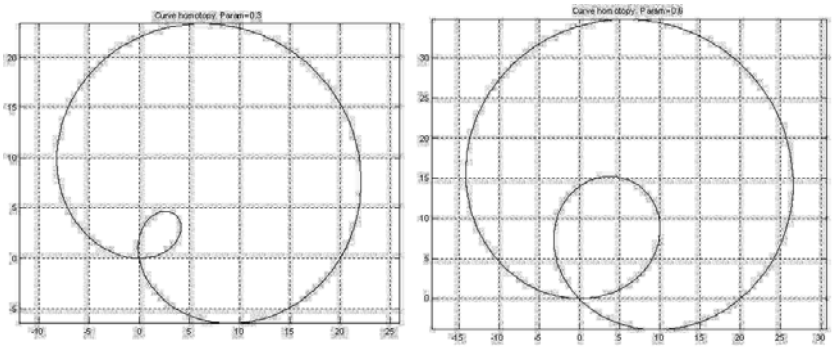

Fig 1. The two images are Homotopic curves (parameter = 0.3 and 0.6 respectively).

It is well known that one of the most important and difficult problem in evolution based applications is the appropriate definition of fitness function. It is this function that determined the selection and optimization of the evolution and the final solution. In the literature, many authors use the artificial selection technique which indeed can solve part of this problem, but delimitates some of the evolutionary nature of the problem and adds some uncertainty caused by human determining.

To solve this problem and for the use of the specific application in this paper, we apply a multi-objective fitness function for optimization. And another feature is that we use a partial ordering technique derived from nonlinear analysis to represent the complicated relationship in candidate solutions from the population. Fig3.2 is a diagram of the functional units of the system.

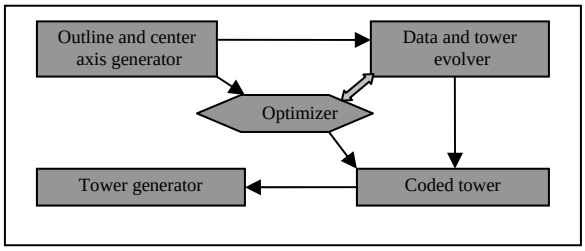

Fig 2. System functional units

4. Genetic algorithms and evolutionary models

Having become widely used for a broad range of optimization problems in the last ten years, the GA has been described as being a "search algorithm with some of the innovative flair of human search". GAs are today renowned for their ability to tackle a huge variety of optimization problems (including discontinuous functions), for their consistent ability to provide excellent results and for their robustness. Natural evolution acts through large populations of creatures which reproduce to generate

new offspring that inherit some features of their parents (because of random crossover in the inherited chromosomes) and have some entirely new features (because of random mutation). Natural selection (the weakest creatures die, or at least do not reproduce as successfully as the stronger creatures) ensures that, on average, more successful creatures are produced each generation than less successful ones. As described previously, evolution has produced some astonishingly varied, yet highly successful forms of life. These organisms can be thought of as good 'solutions' to the problem of life. In other words, evolution optimizes creatures for the problem of life.

Any evolutionary models require architectural concepts to be described in forms of genetic codes. Then these codes are mutated and developed by computer programs into a series of models called population. While models are evaluated in optimization or selection sub-systems, the codes of successful models are constantly picked up until a particular stage of development is selected for prototyping in the real world.

In order to achieve the evolutionary model it is necessary to define the following: a genetic code script, rules for the development of the code, mapping of the code to a virtual model and, most importantly, the criteria for selection. The representation of phenotypes is a fundamental element of any evolutionary system. Phenotypes will define the representation of designs, which will formulate all allowable solutions that can be evolved by the system. Moreover, the phenotype representation plays a significant role in determining the size and complexity of the genotype. The two main representation methods are the surface representations (or boundary representations) and constructive solid geometry (CSG). The first method typically uses combinations of equations and control points to specify shapes, while CSG combines different primitive shapes to form more complex shapes. There are a third of the commonly used solid representations which are called spatial partitioning. This is to decompose a solid into a collection of smaller, adjoining, non-intersecting solids that are more primitive than the original solid. There are a number of variations including: cell decomposition, spatial-occupancy enumeration, octrees, and binary space-partitioning trees.

Primarily, our model in this paper is the boundary representation model. In this representation, towers will be represented by the mathematical data of the main model geometry. Changing the surfaces would be simply achieved by adding or subtracting the mathematical data. The second problem is the data model. In accordance with the mathematical functions type data, we will use the cell division model. A cell division model is based on the structure of a living creature. As in nature, the shape of a living creature is constructed from the basic genetic information to the cells and organisms. The genotype contains information that is the basic construction unit of everything, called the chromosome. Chromosomes will form proteins and other large molecules. Chains of molecules will form tissues and organisms till the whole body. In natural environment, development begins with the chromosome, which forms the base. Then a number of smaller cells are constructed. Large cells are resulted from joining and other operations and form a multi-cellular structure. In a word, the cell division models simple divide the whole into a basic units and operations.

For better description of the combinations of functions, we will use the item jelly model. This is a derivation of the cell model. A layer called the jelly layer is added to represent a compounded structure. We will use this model to represent functions in

various combinations. Basically, the model has three layers, that is, the gene layer, the cell layer and the jelly layer.

The second model in our project is the discrete model. The two basic structures in this model are the section and outline data structure. Each of these two outlines is an ordered set of double numbers, with auxiliary data indicating steps and size. To eliminate the discontinuities caused by data, a mollifier operation is introduced.

Mollifiers act as a function smoother. They can smooth a discontinuous function by substituting the value at one point by some average in a neighbourhood. In the one dimensional case, a mollifier is a nonnegative, real-valued function $J \in C_0^\infty(R)$ such that $J(x) = 0$ if $|x| \geq 1$, and $\int_R J(x)dx = 1$.

5. Particle swarm optimization

Particle swam optimization is a stochastic optimization technique motivated by the behavior of a flock of birds or the sociological behavior of a group of people. There are many improvements to the original PSO. In this section, we briefly introduce the traditional PSO and its variations.

5.1 Basic PSO algorithm

The PSO is a population based optimization technique, where the population is called a swarm. Each particle represents a possible solution to the optimization problem. Unlike evolution process, PSO does not use genetic operations. Instead, particles fly in the n-dimensional search space according to a speed.

Suppose that $x_i = (x_i^1, x_i^2, \cdots, x_i^n)$ is the current position of the particle with index i. We use $v_i = (v_i^1, v_i^2, \cdots, v_i^n)$ to represent the current velocity of particle number i. Let $p_i = (p_i^1, p_i^2, \cdots, p_i^n)$ be the current personal best position of particle i and $f(x)$ be the target function which will be minimized. Now we write the size of the population, that is, the number of all particles, by s, and denote the current global best position found by all particles during previous steps be $p_g(t)$. Therefore, the evolution equation of basic PSO is

$$\begin{cases} v_i(t+1) = \omega v_i(t) + c_1 r_1(t)(p_i(t) - x_i(t)) + c_2 r_2(p_g(t) - x_i(t)) \\ x_i(t+1) = x_i(t) + v_i(t+1) \end{cases} \tag{5.1}$$

Where c_1 and c_2 are the acceleration coefficients, r_1, r_2 are elements from two uniform random sequences in the range (0,1).

5.2 PSO with constriction factor

An alternative version of PSO incorporates a parameter called the constriction factor and the swarm is manipulated according to the equations 12.

$$\begin{cases} v_i(t+1) = \chi(v_i(t) + c_1 r_1(t)(p_i(t) - x_i(t)) + c_2 r_2(p_g(t) - x_i(t))) \\ x_i(t+1) = x_i(t) + v_i(t+1) \end{cases} \tag{5.2}$$

Here χ is the constriction factor, and denote the cognitive and social parameters respectively. It is usually chosen with uniform distribution in the interval [0,1]. The value of the constriction factor χ is typically obtained through the formula $\chi = 2\kappa / \left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|$ where $\phi > 4$, $\kappa = 1$, and $\phi = c_1 + c_2$. Different configurations of χ , as well as a theoretical analysis of the derivation of χ can be found in 12.

5.3 LBest model

Since the original PSO uses global best position, it is called Gbest model. In order to avoid premature convergence, R. C. Eberhart, P. Simpson, and R. Dobbins 13 proposed the Lbest model. Instead of using a unique attractor, the Lbest model uses multiple attractors. This approach divides the population into multiple neighborhoods where each neighborhood maintains a local best position.

The evolution equation is described as in (5.3). Notice that the neighborhood N_i does not related to the actual position of particles in the searching space. Instead, it is only related to the coding, or index position of particles.

$$\begin{cases} N_i = \{p_{i-l+\sigma} \mid \sigma = 0,1,\cdots,2l\},\ x_i(t+1) = x_i(t) + v_i(t+1) \\ p_i(t+1) \in \{N_i \mid f(p_i(t+1)) = \min\limits_{x \in N_i} f(x)\} \\ v_i(t+1) = \omega v_i(t) + c_1 r_1(t)(p_i(t) - x_i(t)) + c_2 r_2(p_g(t) - x_i(t)) \end{cases} \tag{5.3}$$

5.4 Improved PSO

There are many literatures focusing on variations and improvements of the traditional PSO. For the purpose of this paper, we present several aspects of these researches. Firstly in 1998, P.J. Angeline in 14 proposed a generalized PSO based on selection, the tournament selection method. In this paper, an individual is compared with others by fitness value. Next sort the population and replace part of the population with worst fitness by other individuals with better fitness. The procedure of the algorithm is as follows.

(1) Select an individual and compare its fitness value with every other individual. If it is better, then add one score to itself. Repeat this procedure until every individual has a score.

(2) Sort the population according to its score.

(3) Copy half of the population with higher score and replace the other half with lower score.

In another paper 15, P.J. Angeline added reproduction to the traditional PSO. According to a user predefined reproduction probability, some individuals are selected into a crossover pool. Particles in this pool crossover pair-wisely and reproduce the same number of offsprings. Then parents are replaced by offsprings.

The whole number of the population remains changed. The crossover operator is shown as follows.

$$\begin{cases} x_a(t+1) = r_1 x_a(t) + (1-r_1)x_b(t) \\ x_b(t+1) = r_1 x_b(t) + (1-r_1)x_a(t) \end{cases}, \begin{cases} v_a(t+1) = [v_a(t) + v_b(t)]\|v_a(t)\| / \|v_a(t) + v_b(t)\| \\ v_b(t+1) = [v_a(t) + v_b(t)]\|v_b(t)\| / \|v_a(t) + v_b(t)\| \end{cases} \quad (5.4)$$

5.5 Niche technology and PSO

There are already several authors taking efforts to combine niche technology with PSO. One of these efforts is made by P.N. Suganthan 16. This is a variation of the Lbest model but neighbor is defined by space position rather than index position. In each generation, distance between particles is calculated. The maximum distance is marked by $d_{\max}$. For ever pair of particles, the ratio $\|x_a - x_b\| / d_{\max}$ is taken as measure to define the neighbor, where $\|x_a - x_b\|$ denotes the distance between them. The neighbor of each particle is dynamic in the sense that this neighbor grows from a one particle neighbor (the particle itself) to the whole population finally.

In another effort, J. Kennedy 17 discussed effects of neighbourhood topology to the performance of PSO. He proposed several basic neighbourhood topologies: the ring topology, wheel topology and its generalizations.

The most promising technique will be the integration of genetic algorithms with PSO. This technique is achieved with population, or subpopulations, which are evolving and flying simultaneously. In order to keep the genetic operations and the flying operations together, we have two models, i.e., the Genetic PSO, and the Swarm Evolution. Fig 5.1 shows the basic architecture of the two paradigms.

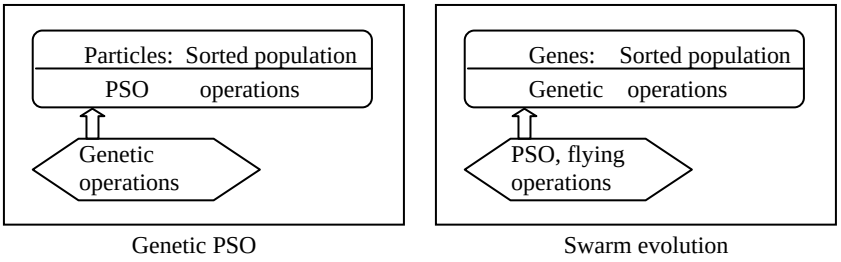

Fig 3. The Genetic PSO and swarm Evolution paradigms.

6. System implementation and generated towers and architectures

Based on the theory described above, we build system implementations based on solid modeling techniques. Two integrated design studio environments have been developed under this project. The platform is personal computers under windows 2000 or windows XP of Microsoft. The programming languages are Microsoft

Visual C++ version 6.0, Acis 3d Kernel, and MatLab version 6.1 of Mathworks. These systems are implemented based on an integration of ACIS 3D solid modeling kernel and MatLab with a C++ graphical user interface. One of the features of the systems is that they are fully compatible with commercial CAAD tools and systems, as well as rapid prototype facilities. A large number of object-oriented components of sophisticated surfaces and envelopes based on a taxonomy of generic form have been built using evolutionary techniques and partial ordering theory, Computational mechanisms have also been developed with which these basic data structures and components can be visualized, combined or split to allow new data structures or new forms to be derived using generative techniques. We are also exploring the possibility to scale up the applications with potentially thousands of solid objects with textual and spatial design details in our Global Virtual Design Studio powered by high performance computer and multiple VR projection facilities.

Now we describe some specific features of the design studio. The first one is named as TowerDev. Features include fully controllable solid modelling environment; nonlinear transform of existing solid models to give new designs; Acis sat viewer; fly through viewer; colour and rendering; programmable design for architecture. The main user interface is as in Fig 6.1.

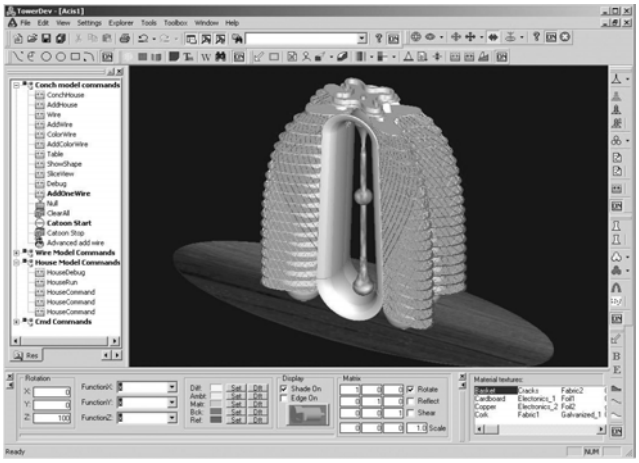

Fig 4. Generated house by TowerDev.

Now we give some more examples of generated towers and other solid models by the studio TowerDev.

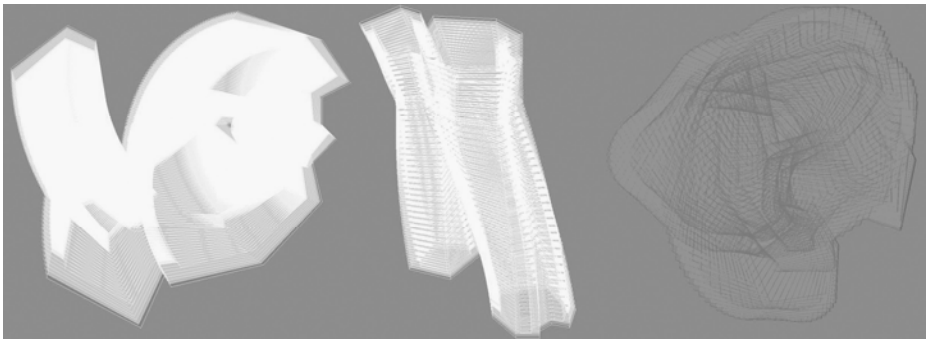

Fig 5. Generated towers with rendering.

7. Acknowledgements

This work is carried out under the “Taishan Scholar” project of Shandong China. It is also supported in part by the Natural Science Foundation of China, the Natural Science Foundation of Shandong Province (No. Z2004G02), and the Scientific and Technology Project of Shandong Education Bureau J05G01.

References

1. Frazer, J. H. (2002) A natural model for architecture —The nature of the evolutionary model. In Cyber Reader, edited by Neil Spiller. Phaidon Press Limited. London, 2002.
2. Frazer, J. H. (2001) Design Workstation on the Future. Proceedings of the Fourth International Conference of Computer-Aided Industrial Design and Conceptual Design (CAID & CD '2001), International Academic Publishers, Beijing, 2001; 17-23.
3. Frazer, J. H. (1995) An Evolutionary Architecture. Architectural Association Publications, London, 1995.
4. Frazer, J. H. (2000) Creative Design and the Generative Evolutionary Paradigm. In P. Bentley ed. Creativity and Design. In press. 2000.
5. Gerd Fischer (editor), Mathematical Models, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, 1986.
6. Tang, M. X. Knowledge-based design support and inductive learning. PhD Thesis, Department of Artificial Intelligence, University of Edinburgh, 1996.
7. Tang, M.X. A knowledge-based architecture for intelligent design support. The Knowledge Engineering Review, 1997;12(4):387-406.
8. Peter J. Bentley (1996), Generic evolutionary design of solid objects using a genetic algorithm. Thesis of doctor of philosophy, University of Huddersfield.
9. Xiyu Liu, Mingxi Tang and John H. Frazer (2002) Shape reconstruction by genetic algorithms and artificial neural networks. Proceedings of The 6th world Multiconference on systemics, cybernetics and informatics (SCI2002).
10. Foley, J., van Dam, A., Feiner, S., Hughes, J. (1990). Computer Graphics Principles and Practice (second edition). Addison-Wesley.
11. Goldberg, D. E., (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley.

12. M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Trans. Evolutionary Computation*, vol. 6, pp. 58–73, Feb. 2002.
13. R. C. Eberhart, P. Simpson, and R. Dobbins, Computational Intelligence PC Tools: Academic, 1996, ch. 6, pp. 212–226.
14. P.J. Angeline, “Evolutionary optimization versus particle swarm optimization: philosophy and performance differences”, in V. William Porto, N. Saravanan, Donald E. Waagen, A. E. Eiben (Eds.): *Evolutionary Programming VII, 7th International Conference, EP98*, San Diego, CA, USA, 1998, pp. 601-610.
15. P.J. Angeline, “Using selection to improve particle swarm optimization”, in *Proc. IEEE World Congress on computational intelligence, ICEC-98*, Anchorange, Alaska, 1998, pp.84--89.
16. P. Suganthan, “Particle swarm optimiser with neighbourhood operator”, in: Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.): *Proceedings of the Congress of Evolutionary Computation*, Vol. 3. IEEE Press, 1999, pp.1958-1962.
17. J. Kennedy, “Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance”, in *Proc. Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Service Center, 1999, pp.1931–1938.
18. R. Kicinger, T. Arciszewski, and KD Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art", Computers & Structures, Vol. 83, No. 23-24, pp. 1943-1978, September 2005.