Generative Art Images by Complex Functions Based Genetic Algorithm

Hong Liu, Xiyu Liu
School of Information Scinece and Engineering, Shandong Normal University, Jinan, P. R. China,250014
School of Management, Shandong Normal University, Jinan, , P. R. China,250014
lhsdcn@jn-public.sd.cninfo.net

Abstract. This paper presents a novel computer supported design system which uses computational approach to producing 3D images for stimulating creativity of designers. It put forward generic algorithm based on a binary tree to generate 3D images. This approach is illustrated by an artwork design example, which uses general complex function expressions to form 3D images of artistic flowers. It shows that approach is able to generate some innovative solutions and demonstrates the power of computational approach.

1 Introduction

The design research community has spent much of its effort in recent years developing computer supported design systems. Generative design systems - systems for specifying, generating and exploring spaces of designs and design alternatives - have been proposed and studied as a topic of design research for many years.

Generative Design is an excellent snapshot of the innovative process from conceptual framework through to specific production techniques and methods. It is ideal for aspiring designers and artists working in the field of computational media. Especially those who are interested in the potential of generative/ algorithmic/ combinational/ emergent/ visual methods and the exploration of active images.

This paper presents a novel computer supported design system that uses evolutionary approach to generate 3D images. The tree structure based genetic algorithms and complex functions are used in this system. Programs are implemented by using Visual C++6.0 and mathematical software MATLAB.

The remainder of this paper is organized as follows. Section 2 is related work on generative design. Section 3 introduces the tree structure based genetic algorithm. In section 4, an artwork design example is presented for showing how to use the tree structure based genetic algorithm and complex functions to generate 3D images.

Finally, these results are briefly analyzed, followed by a discussion of future improvements.

2 Related work

Generative design is the idea realized as genetic code of artificial objects. The generative project is a concept-software that works producing three-dimensional unique and non-repeatable events as possible and manifold expressions of the generating idea identified by the designer as a subjective proposal in a possible world. This idea/human creative art renders explicit and realizes an unpredictable, amazing and endless expansion of human creativity. Computers are simply the tools for its storage in memory and execution [1].

Generative design describes a broad class of design where the design instances are created automatically from a high-level specification. Most often, the underlying mechanisms for generating the design instances in some way model biological processes: evolutionary genetics, cellular growth, etc. These artificial simulations of life processes provide a good conceptual basis for designing products.

Evolving design concepts by mutating computer models in a simulated environment is now a well-established technique in fields as diverse as aeronautics, yacht design, architecture, textile design, fine art and music [2]. Some of the work was performed by Professor John Frazer, who spent many years developing evolutionary architecture systems with his students. He showed how evolution could generate many surprising and inspirational architectural forms, and how novel and useful structures could be evolved [3]. In Australia, the work of Professor John Gero and his colleagues also investigated the use of evolution to generate new architectural forms. This work concentrates on the use of evolution of new floor plans for buildings, showing over many years of research how explorative evolution can create novel floor plans that satisfy many fuzzy constraints and objectives [4]. They even show how evolution can learn to create buildings in the style of well-known architects.

In Argenia, a system for architectural design by Soddu, the three-dimensional models produced can be directly utilized by industrial manufacturing equipment like numerically controlled machines and robots, which already represent the present technologies of industrial production. This generative and automatic reprogramming device of robots makes it possible to produce unique objects with the same equipment and with costs comparable to those of objects that are identical; like a printer that can produce pages that are all the same or all different, at precisely the same cost [5].

The list of experimental application of generative techniques to physical design also includes airfoil optimization [6], clothing design [7], and an experimental test-bed for consumer controlled generative product design by Emergent Design [8].

However, the development of generative design tools is still at its early stage. The research and development of design support tools using evolutionary computing technology are still in process and have huge potential for the development of new design technology.

3 The tree structured genetic algorithm

General generic algorithms use binary strings to express the problem. It has solved many problems successfully. But it would be inappropriate to express flexible problem. For example, mathematical expressions may be of arbitrary size and take a variety of forms. Thus, it would not be logical to code them as fixed length binary strings. Otherwise the domain of search would be restricted and the resulting algorithm would be restricted and only be applicable to a specific problem rather than a general case. Thus, tree structure, a method useful for representing mathematical expressions and other flexible problems, is presented in this paper.

For a thorough discussion about trees and their properties, see [9,10]. Here, we only make the definitions involved in our algorithm and these definitions are consistent with the basic definitions and operations of the general tree.

Definition 1 A binary complex function expression tree is a finite set of nodes that either is empty or consists of a root and two disjoint binary trees called the left sub-tree and the right sub-tree. Each node of the tree is either a terminal node (operand) or a primitive functional node (operator). Operand can be either a variable or a constant. Operator set includes the standard operators (+, -, *, /, ^) , basic mathematic functions (such as sqrt (), exp( ), log( )), triangle functions (such as sin(x), cos(x), tan(x),cot(x),sec(x),csc(x),asin(x),acos(x)), hyperbolic functions (such as sinh( ), cosh( ), tanh ( ), asinh ( ), acosh( ), atanh( )), complex functions (such as real(z), imag(z), abs(z), angle(z), conj(z)) and so on.

Here we use the expression of mathematical functions in MATLAB (mathematical tool software used in our system).

A binary complex function expression tree satisfies the definition of a general tree:

There is a special node called the root.

The remaining nodes are partitioned into $n \geq 0$ disjoint sets, where each of these sets is a tree. They are called the sub-tree of the root.

Genetic operations include crossover, mutation and selection. According to the above definition, the operations are described here. All of these operations take the tree as their operating object.

(1) Crossover

The primary reproductive operation is the crossover operation. The purpose of this is to create two new trees that contain ‘genetic information’ about the problem solution inherited from two ‘successful’ parents. A crossover node is randomly selected in each parent tree. The sub-tree below this node in the first parent tree is then swapped with the sub-tree below the crossover node in the other parent, thus creating two new offspring. A crossover operation is shown as figure 1.

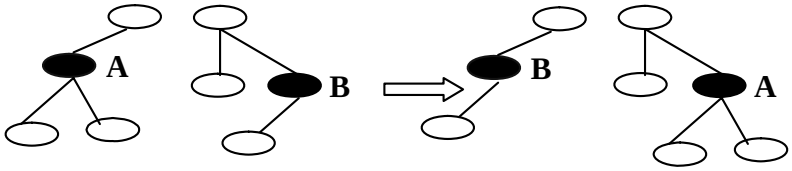

Fig. 1. A crossover operation

(2) Mutation

The mutation operation is used to enhance the diversity of trees in the new generation thus opening up new areas of 'solution space'. It works by selecting a random node in a single parent and removing the sub-tree below it. A randomly generated sub-tree then replaces the removed sub-tree. A mutation operation is shown as figure 2.

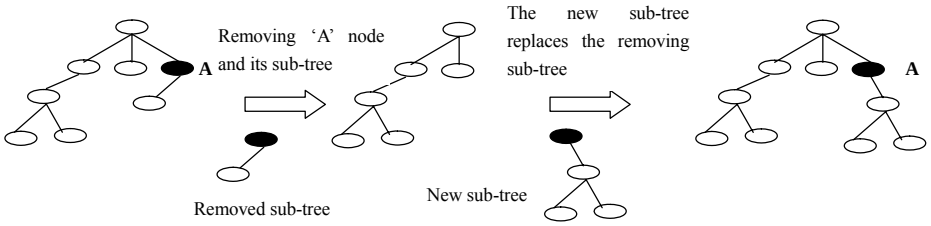

Fig. 2. A mutation operation

(3) Selection

For general design, we can get the requirement from designer and transfer it into goal function. Then, the fitness value can be gotten by calculating the similar degree between the goal and individual by a formula. However, for creative design, it has no standards to form a goal function. Therefore, it is hard to calculate the fitness values by a formula. In our system, we use the method of interaction with designer to get fitness values. The range of fitness values is from -1 to 1. After an evolutionary procedure, the fitness values that appointed by designer are recorded in the knowledge base for reuse. Next time, when the same situation appears, the system will access them from the knowledge base[11].

This method gives the designer the authority to select their favored designs and thus guide system to evolve the promising designs. Artificial selection can be a useful means for dealing with ill-defined selection criteria, particularly user centered concerns.

Many explorative systems use human input to help guide evolution. Artists can completely take over the role of fitness function. Because evolution is guided by human selectors, the evolutionary algorithm does not have to be complex. Evolution is used more as a continuous novelty generator, not as an optimizer. The artist is likely to score designs highly inconsistently as he/she changes his/her mind about desirable features during evolution, so the continuous generation of new forms based on the fittest from the previous generation is essential. Consequently, an important element of the evolutionary algorithms used is non-convergence. If the populations

of forms were ever to lose diversity and converge onto a single shape, the artist would be unable to explore any future forms.

For clarity, we will present the performing procedure of the tree structured generic algorithms together with a flowers generative design example in the next section.

4 A generative artwork example

An artwork design example is presented in this section for showing how to use tree structure based generic algorithm and complex function expressions to generate 3D images.

The complex function expressions are used to produce 3D artistic images. Here, z=x+iy (x is real part and y is virtual part), complex function expression f(z) is an in-order traversal sequence by traversing complex function expression tree.

Both real, imaginary parts and the module of f(z) can generate 3D images by mathematical software MATLAB. Three images of f(z)=sin(z)*log(-z^2)*conj(z) are shown as figure 3.

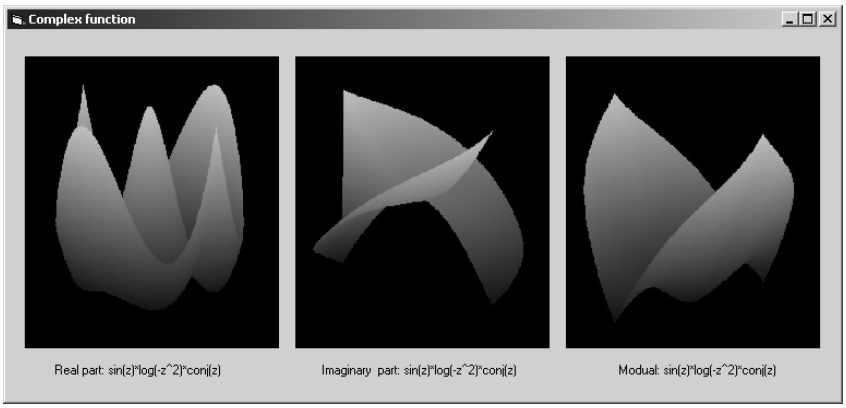

Fig. 3. Three images of f(z)=sin(z)*log(-z^2)*conj(z)

Next, we will present the performing process of the algorithm step by step.

Step 1: Initialize the population of chromosomes. The populations are generated by randomly selecting nodes in the set of operands and the set of operators to form complex function expressions. We use the stack to check whether such a complex function expression has properly balanced parentheses. Then, using parsing algorithm, the complex function expressions is read as a string of characters and the binary complex function expressions tree is constructed according to the rules of operator precedence.

Step 2: Get the fitness for each individual in the population via interaction with designer. The populations with high fitness will be shown in 3D form first. Designer can change the fitness value when they have seen the 3D images.

Step 3: Form a new population according to each individual's fitness.

Step 4: Perform crossover and mutation on the population.

Figure 4 shows two complex function expressions trees. Their expressions are f(z)=1.5*sin(z)*cos(-z^2)*angle(-z)*exp(-z) and f(z)=sqrt(z)*log(-z^2)*cot(-z) respectively.

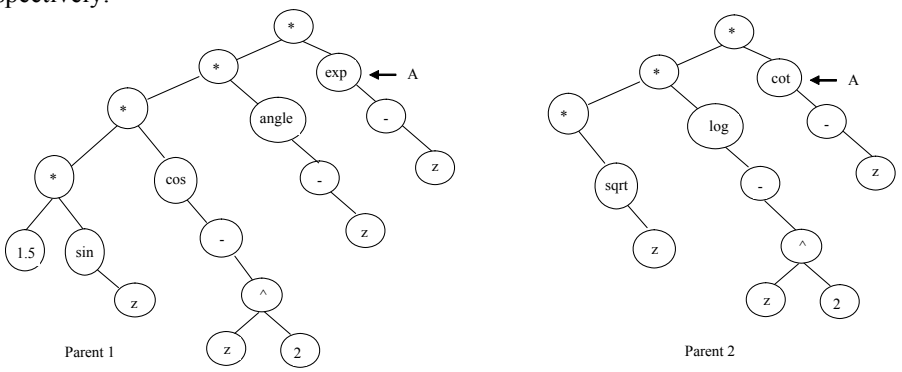

Fig. 4. Two parent trees with one crossover nodes

(1) Crossover operation

A crossover node is randomly selected in each parent tree. The sub-tree below this node on the first parent tree is then swapped with the sub-tree below the crossover node on the other parent, thus creating two new offspring. If the new tree can't pass the syntax check or its mathematical expression can't form a normal sketch shape, it will die.

Taking the two trees in figure 4 as parent, after the crossover operations by nodes 'A', we get a pair of children (see figure 5).

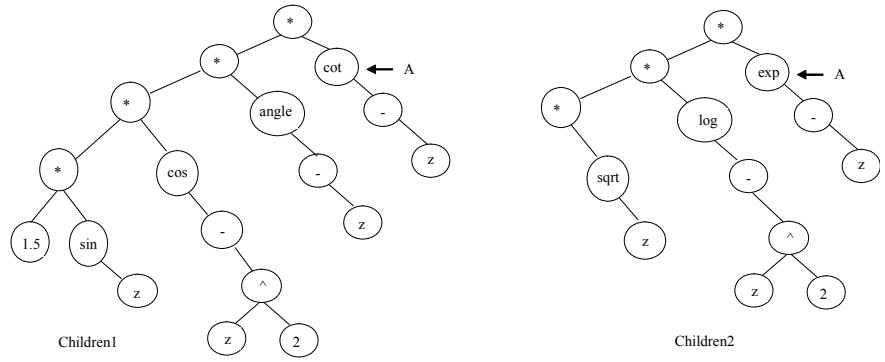

Fig. 5. The two children generated by a crossover operation

Figure 6 and figure 7 shows a group of generated 3D images by the module and imaginary part of f(z) correspond to figure 4 and figure 5.

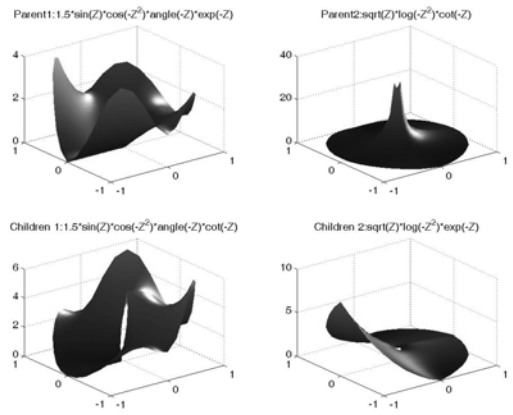

Fig. 6. The images correspond to the module of f(z) in figure 4 and figure 5

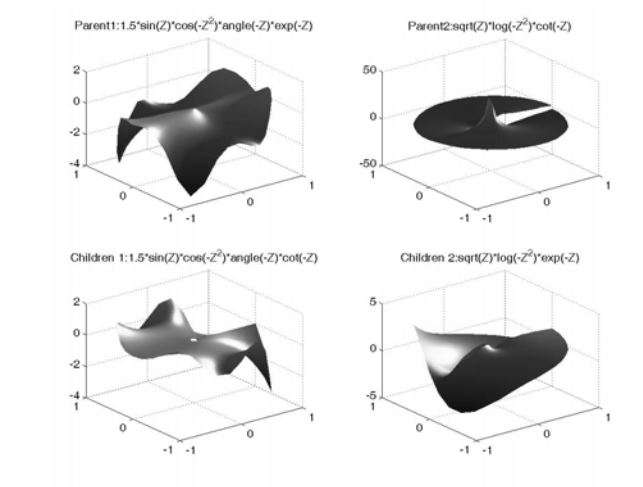

Fig. 7. The images correspond to the imaginary part of f(z) in figure 4 and figure 5

(2) Mutation operation

The mutation operation works by selecting a random node in a single parent and removing the sub-tree below it. A randomly generated sub-tree then replaces the removed sub-tree. The offspring will die if it can't pass the syntax check or it can't form a normal shape.

Taking the parent1 tree in figure 4 as a parent, one offspring generated by mutation operation is shown as figure 8. In which, child is generated by replacing node A and its sub-tree with new sub-tree. Figure 9 is the images correspond to the imaginary part of f(z) in figure 8.

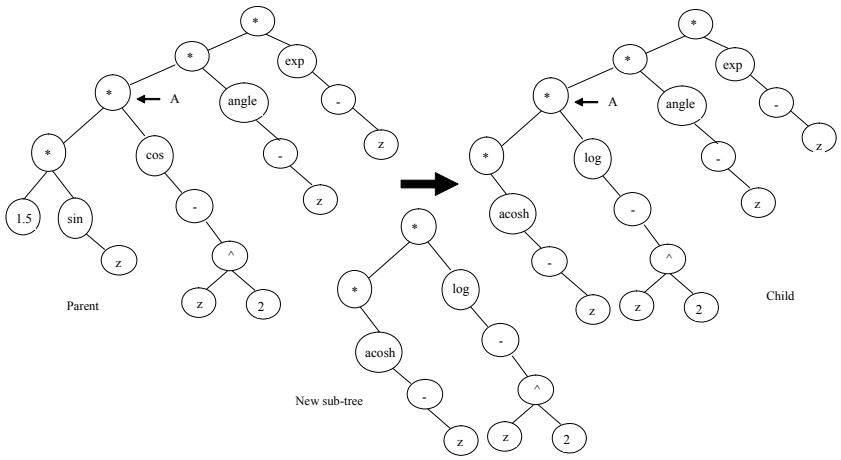

Fig. 8. The results of one mutation operation

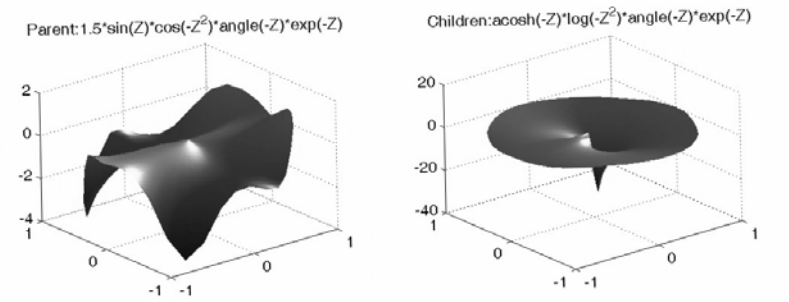

Fig. 9. The images correspond to the imaginary part of f(z) in figure 8

Step 5: If the procedure doesn't stopped by the designer, go to step 2.

This process of selection and crossover, with infrequent mutation, continues for several generations until it is stopped by the designers. Then the detail design will be done by designers with human wisdom.

The generated images are handled by designers using computer operations, such as rotating, cutting, lighting, and coloring and so on. The interactive user interface can be seen in figure 10. Then, we can get some artistic flower images as shown in figure 11.

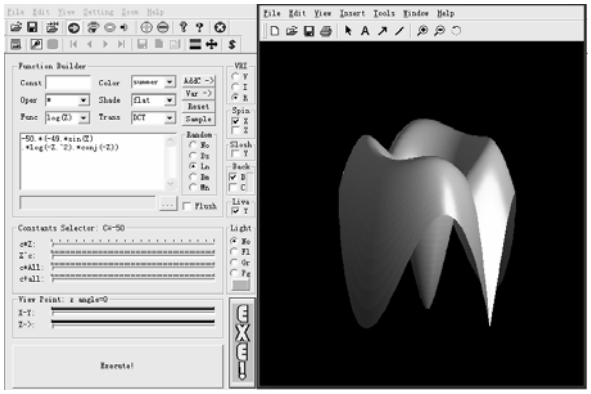

Fig. 10. The interactive user interface

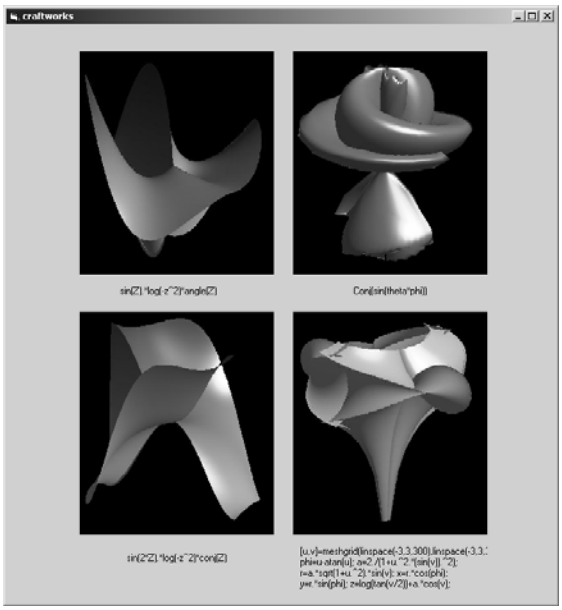

Fig. 11. Some artistic flowers generated by complex function expressions

5 Conclusions

Generative design is still in its infancy. There is still much work to be done before the generative design system can be in practice. Our current work is to use the multi-agent architecture as an integrated knowledge-based system to implement a number of generative design techniques including genetic algorithms and fractal

geometry. These new algorithms will then be fully integrated with a selected set of 2D (sketching) and 3D (surface and solid modeling) tools and other design support systems. This integrated system is intended for supporting creative design in a visual environment [12].

This project is founded by National Natural Science Foundation of China (No. 69975010, No. 60374054) and supported by Natural Science Foundation of Shandong Province (No. Z2004G02, Z2006G09).

References

1. C. Soddu. Argenia, naturality in industrial design, In: Proc. 3rd International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID&CD'2000, International Academic Publisher, World Publisher Corporation, December 2000.
2. R. Saunders, J.S. Gero. Artificial creativity: A synthetic approach to the study of creative behaviour, in JS Gero and ML Maher (eds), Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, 2001,pp. 113-139.
3. J. H. Frazer etc. Generative and evolutionary techniques for building envelope design .Generative Art 2002.
4. J. S. Gero, V. Kazakov. An exploration-based evolutionary model of generative design process. Microcomputers in Civil Engineering 1996, 11:209-216.
5. C. Soddu,Visionary variations, personal exhibitions at Hong Kong Museum, Visual Art Centre, 2002.
6. I. DeFalco, R. DelBalio,A. DellaCioppa,E Tarantino, A parallel genetic algorithm for transonic airfoil optimisation, In: Proc. International Conference on Evolutionary Computation 1995, Perth, Western Australia, 1995.
7. Y. Nakanishi, Capturing preference into a function using interactions with a manual evolutionary design aid system. In: Proc. Genetic Programming '96, Stanford University, 1996.
8. M. Pontecorvo. N. Elzenga, Exploring designer/consumer dialog in evolutionary product design systems, In: Proc. AISB'99, Society for Artificial Intelligence and Simulated Behavior, Edinburgh, Scotland, 1999.
9. A. S. Thomas, Data Structure, algorithms, and software principles, Addison-Wesley Publishing Company, inc.U.S.A. 1994.
10. B. Mckay, M. J. Willis, G. W. Barton, Using a ree structured genetic algorithm to perform symbolic regression, In Proceedings of the First IEE/IEEE International Conference on Genetic Algorithm in Engineering Systems: Innovations and Applications, Halifax Hall, University of Sheffield, UK. 1995, 487-498.
11. H. Liu, M. X. Tang, J. H. Frazer. Supporting dynamic management in a multi-agent collaborative design system. International Journal of Advances in Engineering Software, 2004, 35(8-9): 493-502.
12. H. Liu, M. X. Tang, J. H. Frazer, Supporting creative design in a visual evolutionary computing environment. International Journal of Advances in Engineering Software, 2004, 35(5): 261-271.