International Journal of Pure and Applied Mathematics
Volume 118 No. 18 2018, 2663-2669
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

Generating Innovative Origami Designs using Genetic Algorithms

Ananthalakshmi Sankar
Dept. of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
anu.294@gmail.com

Bharathi D.
Dept. of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
d_bharathi@cb.amrita.edu

Dikshita G
Dept. of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
dikshita.gn@gmail.com

Hari Prasad .P
Dept. of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
hariprasad496@gmail.com

Sai Krishna .R
Dept. of Computer Science and Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
India
krishnasudhamusu@gmail.com

***Abstract*— There are multiple combinations of folds that can be performed on a paper and thus several origami designs are possible in the creative space of design structures, but not all of them have been discovered. This mainly is due to the lack of understanding of the geometry and mathematics of folding structures and the multitude of possibilities for generating such designs. Computational Origami has enabled to model crease patterns and represent designs such that modification and purposeful or technical folding of patterns are made easier. But most of these methods require modeling of the crease pattern to arrive at the origami design desired. This paper introduces "Neori", the extension module of Oripa, by applying Genetic algorithm, representing these crease patterns as genotypes and setting constraints to retain valid structures, to search for innovative flat-foldable origami. The objective function and the evolutionary operators have been formulated to increase complexity of the structures.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Origami is the ancient Japanese art of paper folding which has been developed for centuries to make intricate designs. Generally, the structures are folded out of a square piece of paper and are used in several engineering practices such as packaging, compressing and in miniaturization, such as in

stent graphs for medical usage. Computational Origami is a relatively new field wherein algorithms have been studied for forming folded structures from crease patterns. Robert Lang has done pioneering work in this field and the principles defined in the process of computerizing origami structures has been used by several engineers and artists to design new models. Yet, there are several new and innovative designs in the creative space yet to be discovered. The objective of this paper is to address the complexity involved in Origami Design by generating arbitrary crease patterns with certain design properties.

A. *Computational Origami*

In the late 20th century, principles of geometry which have been applied to origami and mathematical axioms determining foldability were programmed. The operations that can be applied while folding a paper were set initially by Huzita-Hatori axioms and adopted in origami software thereafter. Folds are classified in accordance to lines and points that form the creases for the design [1]. These lines are classified as mountain or valley types of folds, which intersect to form polygonal faces in between. Mathematical principles when applied to such structures have helped introduce Software such as Tree Maker by Robert Lang [2], Oripa by Jun Mitani, Origamizer [3] and Freeform Origami. The former two, being open source software, are convenient to build upon and add new modules such as ours for creating entirely new designs.

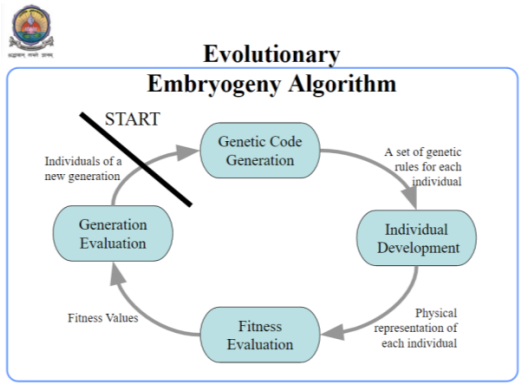
The basis of most origami designs is flat folded structures, which means they can be represented as 2D figures and can be modified and sculpted later [4]. The mathematics of such origami is defined based on some universal results that the lines in the crease pattern must hold in order to ensure assigned flat-foldability.

B. *Genetic Algorithm*

A solution to a problem can be arrived at by either logically inferring a single solution using deductive logic or by arbitrarily arriving at a conclusion using heuristic search. The former has been implemented as circular packing algorithm or mathematical conjectures based on relation between mountain and valleys as in [5]. An evolutionary approach in addition to this folding problem can bring us closer to designing origami without encountering NP completeness [13].

Genetic algorithm performs heuristic searches that imitate the natural selection process as in Darwin's theory [6]. It is done by mixing up parts of individual solutions which results in the speed up of the overall process (crossover operator in EA literature), and with the addition of an amount of stochastic or randomness (mutation operator in EA literature) the algorithm property of escaping local extreme unfolds. This property of the genetic algorithms is known as the adaptation property. In our case, the origami needs to be assumed as the individual that shall evolve as per the selection process. One might assume the folded origami structure to be the individual, but it is easier to represent the gene in the form of the crease pattern, containing all the lines along which folds are performed in terms of the vertices. Therefore, origami models will be suitably developed into individuals that shall be represented as a genotype string. Figure 1 represents the life cycle of a population when processed by genetic algorithm. These strings will be altered by genetic operators and the newly formed population and the old one shall be evaluated by the fitness function. This process ultimately results in a new generation of chromosome which would most probably be fitter than the initial population

Figure 1. A cycle for new population generation



The main problem that one might face in implementing genetic algorithms to origami structure is an efficient representation of the genetic code, such that even after mutating it, the candidate solution represented by the code still remains valid. This is addressed in the following section along with the appropriate section method, constraints for crossover and mutation, and fitness functions.

II. Design Methods

Patterns for origami designs have been achieved because of several effective computational methods developed. These methods may not be very versatile and may not explicitly mention the fold sequence but nonetheless are appropriate for designing structures based on different types of inputs. Neori looks to integrate with these existing software(s), to evolve variety of new designs.

TABLE I Comparison Of Computational Origami Tools

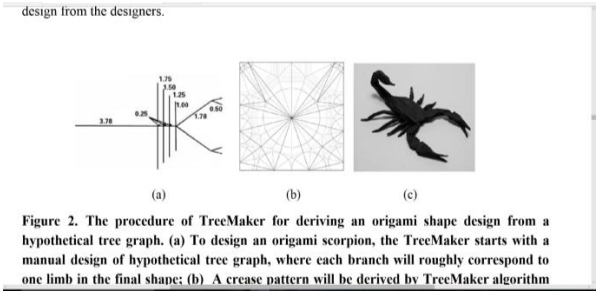
NAME OF SOFTWARE	OBJECTIVE	ENVIRONMENT
Freeform Origami	To interact with origami forms while altering the crease pattern of the model.	Proprietary software
Origamizer	To generate a crease pattern that folds into a given polyhedron	Proprietary software
Tree Maker	Designs a base or folds into a shape for the given stick figure	FOSS: C++
Oripa	To design the crease patterns of Origami and calculate the folded shape from crease pattern	FOSS: Java

A. *TreeMaker and Origamizer*

The basis of many complex origami designs is a geometric shape called a base, which is a folded configuration of the original square that has the same number of flaps of the same length as the appendages of the subject. The TreeMaker

requires the designer to input an approximate tree graph. Depending on the projections or the limbs of this graph the Treemaker algorithm is applied to generate a crease pattern which can then be folded [7]. This program makes as C++ as its source code language and is open source and therefore is a viable option for the implementation of Neori. But it has more sophisticated methods for generation of the crease pattern than for folding it to be graphically represented.

Figure 2. Sequence of treemaker for deriving an origami from a user provided stick figure to folded form, in this case scorpion

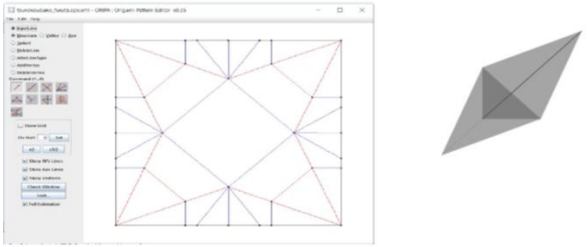


Tachi presented "Origamizer" which used the sheet to form the surface mesh of the desired origami structure [8] unlike Treemaker that required a graph to build the body of the figure. This process of surface division appended with the concept of tucking molecules forms a plane with the connected facets of the structure surface. This software has a good success rate but is a proprietary software and is not feasible for implementation of Neori.

B. Oripa as a Base Software

Neori requires a reliable base software which can be a part of the feasibility function, have accurate geometric representation and reliably transform crease pattern to graphical origami structures [8]. Oripa is a drawing tool with clean rendering of flat foldable origami designs. It takes crease pattern with lines assigned as mountain or valley type as input and after performing foldability checks and operations, it outputs the folded form [10].

Figure 3. UI of Oripa and its folded form graphics



Following are the reasons for Oripa to be chosen as the base software:

1) It is an Open Source software
2) It uses Java as its source code language
3) It has a refined graphical interface comprising of rendering and shading

Oripa uses the mathematical approach to origami by implementing Kawasaki and Maekawa's theorems for checking for foldability. Therefore, this enables ease in reading crease patterns, rendering them and in checking for feasibility with foldability as a parameter.

The input crease patterns are stored in .opx file type which is very similar an xml file. This file stores the values of the lines in the form of coordinates which represent inner vertices on the base.

III. SYSTEM ARCHITECTURE

Primarily the system consists of 5 phases, starting from the initial candidates that are provided as input, to the final solution set of individuals

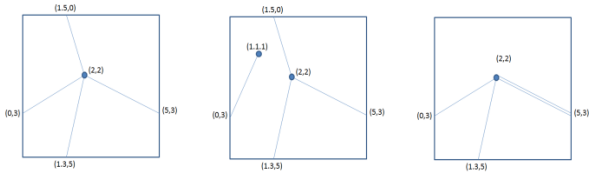
A. Input Crease Pattern

The initial population is presented as .opx files which are extracted using XMLDecoder package. This generates all the edges denoted by their end coordinates and extracts number of creases. Coordinates are of double type and in the form (x,y).

B. Genotype Representation

Initially the coordinates will be of the form {((x1, y1)(x2, y2)), ((x3, y3)(x4, y4)), ((x5, y5)(x6, y6))…..}. This form will be difficult to manage as it will generate invalid crease pattern upon mutation. Figure shows crease1 is a valid crease represented as {((0,3)(2,2)), ((1.5,0)(2,2)), ((1.3,5)(2,2)), ((2,2)(5,3))}. Say mutation causes (x2, y2) to change to (1.1,1) then the pattern crease2 is invalid as it has formed a dangling point. Also if (x3, y3) mutates to (5,3) then the folds overlap, once again causing an invalid crease3.

Figure 4. Genetic Mutation implication



Therefore we separate all the unique coordinates as gene1 and the relations to form edges as gene2 ie gene1{(0,3),(2,2), (1.5,0), (1.3,5),(5,3)} and gene2{(0,1) (2,1) (3,1) (4,1)}

C. Gene Encapsulation

The genotype representation consists of several datatypes and collections. We have encapsulated it into a class structure as follows.

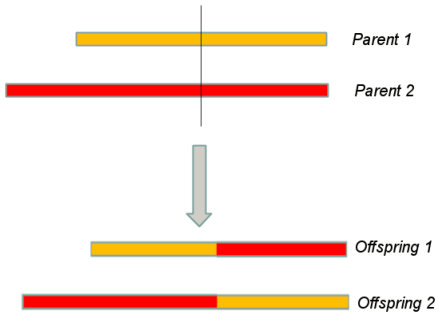
```
class genes{
      CreasePattern pattern;
      double gene1[][];
      int gene2[][];
      int coords;
      byte type[];
      OriLineProxy[] cp1;
      double[][] cp2;
      int nol;
}
```

1. Gene1[ ][ ] (double) : The coordinates of vertices are stored in this data member.
2. Gene2[][ ] (int): The coordinate pair forming the lines are stored in this data member.
3. Coords (int): The number of vertices the crease pattern has is stored here.
4. Type [](byte) : The line type : (i)Mountain (ii)Valley (iii)Auxiliary is represented
5. Cp1 (OriLineProxy[]): The crease pattern is stored as an array of class OriLineProxy which in turn encapsulates the Edge and Vertices class.
6. Cp2[ ][ ] (double): The crease pattern is broken down into lines and coordinates from cp1 and storedin this data member. Using this data member, gene1 and gene2 are constructed.
7. Nol (int): The number of lines present in the crease pattern is stored in this data member.
8. Pattern (CreasePattern): The lines of crease pattern are abstracted and stored on the whole as a CreasePattern object which includes the document properties for graphical representation.

D. Evolutionary Operations

1. Crossover: The population is split into pairs by halving it. Among these pairs each individual chromosome is split into two sets of genes. The lower halves of the gene sets are exchanged to form two new individuals. New generation of fourteen new individuals are created. In case of one gene lengths differing, the crossover will take place for the length of the shorter gene.

Figure 5. Crossover of gene



2. Mutation: A point from the crease pattern is chosen at random and its value is changed. The association of the lines with the point is not altered. This helps in retaining the validity of the crease pattern.
3. Duplicates removal: After crossover and mutation, there are chances of getting duplicates of the same point. The duplicates are found and altered until there are no more repetitive points.

E. Fitness Evaluation

1. Foldability: The generated crease pattern may not adhere to the axioms of origami. This necessitates the need to check their validity. For a flat folded origami to be valid the following conditions must apply:
   - Total number of creases is even. ie M + V = 2n
   - The difference between the count of mountain and valley folds is ±2. Ie M - V = ±2
   - The alternate angles about the vertex sum upto π

   The inner vertices of the crease pattern must satisfy all of the above conditions. If any one of these conjunctures is not applicable then the pattern is not a flat foldable origami. [14]
2. Induced Foldability: Due to crossover and mutation the chances of the crease getting invalidated is very high. Once that happens, the crease patterns are modified to form a valid one. This increases the chances of retaining the crease pattern even though it does not guarantee success in all cases.
3 Complexity: The complexity of the pattern is taken as a parameter here because it increases the creativity index of the patterns. Except for the first run, in all other runs there will be two sets of generations i.e. twenty-eight crease patterns to choose from. The sum of the lines forming the crease pattern is calculated. The sum is directly proportional to the complexity of the crease pattern. The individuals with the highest complexity are chosen to form the next generation.

F. Regeneration of Crease Pattern

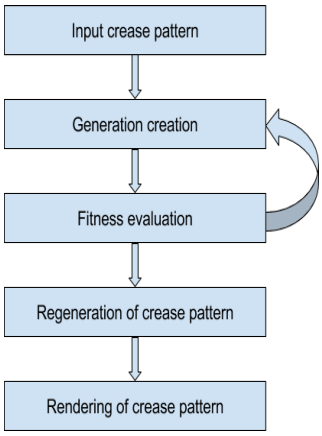
The crease pattern at present is in the form of our chromosomal structure. This is performed by going up the

various levels of extraction and finally the crease pattern is converted to .opx form.

G. Rendering of Crease Pattern

These .opx format crease patterns are then displayed in their graphical form by applying generation of folding sequence algorithms in Oripa. This form also depicts the depth of the layers through gradients in shading.

Figure 6. Flowchart of the system



IV. Implementation and Validation

The basic GA framework is followed wherein all necessary parameters were set.

Size of origami paper = [-200,-200] x [200,200]
Initial population = 14 individuals
Maximum number of lines in a crease = 600
Maximum coordinates assumed = 200
Number of generations = 20
Selection mechanism used is Ranking

Based on the number of generations run, we have observed that the complexity of the origami design increase. For instances, since our fitness function preferred designs with more number of lines, the final designs will have maximum number of creases.

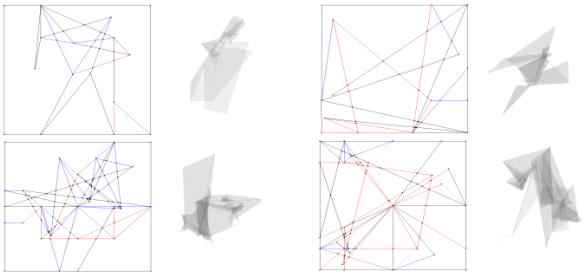
A different complex design is being generated on each run as see in Table II which considers any 5 of the population. Although the number of lines changes it is always much greater than complexity of initial population.

TABLE II COMPLEXITY COMPARISON OF THE INITIAL AND FINAL POPULATION OF ANY 5

INDIVIDUAL NO.	INITIAL POPULATION NOL	FINAL POPULATION NOL		
		ITER 1	ITER 2	ITER 3
A	48	176	240	272
B	152	570	583	418
C	24	110	32	54
D	75	120	107	164
E	132	468	422	350

Figure 6 is some of the sample output structures along with their crease patterns. Each structure is unique and largely complex. An upper limit has been put on the number of lines to restrain the complexity from going out of bonds such that it becomes beyond the capability of the designer to fold the pattern. Although foldability constraints largely reduce the number of possible designs we still have several foldable outputs, thus making our system valid for further development and use.

Figure 6. Output Origami design and their Crease Patterns



V. Conclusion

This research proposed a solution to the design problem in origami that had till date been following a linear approach. Unlike TreeMaker software Neori can suggest several designs if the constraints or desired properties of the structure can be briefly described. For instance, if we want innovative designs for flat topped furniture like a table, chair or desk, we only need to set the constraint as flat surface area to be maximum or number of overlapping folds to be maximum (to make the structure sturdy). The constraints and the selection method applied can be customized to achieve a variety of designs of desired properties.

Presently we have successfully implemented a model to increase complexity in terms of number of folds by incorporating GA with the existing pattern folding tool, Oripa. Therefore, we have implemented a system wherein for the first-time innovative designs are being generated by evolutionary methods.

References

[1] Demaine E. D., O'Rourke J.: Geometric Folding Algorithms. Cambridge University Press, 2007.

[2] Robert J. Lang. TreeMaker 4.0: A Program for Origami Design, 1998. http://origami.kvi.nl/ programs/TreeMaker/trmkr40.pdf.

[3] Tachi, T. "3D Origami Design based on Tucking Molecule." Fourth International Meeting of Origami Science, Mathematics, and Education, Pasadena, CA, 2006.

[4] T. Hull. On the mathematics of at origamis. Congressus Numerantium 100 (1994).

[5] M. Bern, B. Hayes: The complexity of flat origami. In Proceedings of the 7th ACMSIAM . Symposium on Discrete Algorithms, pp. 175–183, 1996.

[6] Holland, John H. Genetic Algorithms, Scientific American, pp. 66-72, 1992.

[7] Lang, R. J. "A Computational Algorithm for Origami Design", 12th Annual ACM Symposium on Computational Geometry, Philadelphia, PA. 1996

[8] Tachi, T. "Simulation of Rigid Origami." Fourth International Meeting of Origami Science, Mathematics, and Education, Pasadena, CA, 2006.

[9] Tachi, T. "Origamizing Polyhedral Surfaces," IEEE Trans. Visual. Computer. Graphics, 16(2), pp. 298–311, 2010.

[10] Hugo Akitaya, Jun Mitani, Yoshihiro Kanamori and Yukio Fukui, "Generating Folding Sequences from Crease Patterns of Flat-Foldable Origami", ACM SIGGRAPH 2013 Posters, Anaheim, CA. 2013.

[11] Mitani, J. Development of origami pattern editor (ORIPA) and a method for estimating a folded configuration of origami from the crease pattern. IPSJ Journal 48, 9 (sep), 3309–3317, 2007.

[12] Fumiaki Kawahata, "The technique to fold free angles of formative art 'origami'," abstract in proceedings of The Second International Meeting on Origami Science and Scientific Origami, Otsu, Japan, 1994. M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.

[13] T. Stanković, M. Stošić and D. Marjanović , "Evolutionary Algorithms in Design", International Design Conference - Design 2006 Dubrovnik - Croatia, May 15 - 18, 2006.

[14] Jun Mitani, The Folded Shape Restoration And The Rendering Method Of Origami From The Crease Pattern, In proceedings of the 13th International Conference On Geometry And Graphics, August 4-8, Dresden, 2008.

[15] Langer, M. S., and Bülthoff, H. H. Perception of shape from shading on a cloudy day. Tech. Rep. 17, Max-Planck-Institut für biologische Kybernetik, 1999.

[16] Furuta, Y., Kimoto, H., Mitani, J., and Fukui, Y. Computer Model and Mouse Interface for Interactive Virtual Origami Operation. IPSJ Journal, Vol.48, No.12, pp.3658-3669, 2007.

[17] Lang, R. J. 2012. Origami Design Secrets: Mathematical Methods for an Ancient Art. Second Edition. CRC Press.

[18] Arkin, E. M., Bender, M. A., Demaine, E. D., Demaine, M. L., Mitchell, J. S., Sethia, S., and Skiena, S. S. 2004. When can you fold a map? Computational Geometry 29, 23–46.

[19] Kuribayashi, K., Tsuchiya, K., You, Z., Tomus, D., Umemoto, M., Ito, T., and Sasaki, M. 2006. Self deployable origami stent grafts as a biomedical application of ni-rich tini shape memory alloy foil. Materials Science and Engineering: A 419, 12, 131 – 137.

[20] nishiyama, Y. 2012. Miura folding: Applying origami to space exploration. International Journal of Pure and Applied Mathematics 79, 2, 269–279.