Interactive Genetic Algorithms and their Application in Fashion Design

Nazanin Alsadat Tabatabaei Anaraki, B.Sc
Graduate Faculty of Texas Tech University in
Partial Fulfillment of the Requirements for
the Degree of Master of Science
2500 Broadway, Lubbock, TX 79409, USA

***Abstract* - These days, consumers can make their choice from a wide variety of clothes provided in the market; however, some prefer to have their clothes custom-made. Since most of these consumers are not professional designers, they contact a designer to help them with the process. This approach, however, is not efficient in terms of time and cost and it does not reflect the consumer's personal taste as much as desired. This study proposes a design system using interactive genetic algorithm (IGA) to overcome these problems. IGA differs from traditional genetic algorithm (GA) by leaving the fitness function to the personal preference of the user. The proposed system uses user's taste as a fitness value to create a large number of design options, and it is based on an encoding scheme describing a dress as a two-part piece of clothing. The system is designed in the rhinoceros 3d software, using python, which provides good speed and interface options. The assessment experiments with several subjects indicated that the proposed system is effective.**

INTRODUCTION

Fashion design is an industry closely connected to our lives and it is one of the most important representations of human civilization. People used to either make their own clothes or buy them from small local producers. However, the Industrial Revolution enabled the shift to mass production. In order for the clothing industry to compete with other industries, it should utilize modern technologies.

Since most consumers are not professional designers, those who desire custom-made clothing contact a designer to help them with the process. This approach, however, is not efficient in terms of time and cost, and it often does not reflect the consumer's personal taste as much as desired.

This paper proposes a design system using the Interactive Genetic Algorithm (IGA) to overcome these problems. The system is designed in the Rhinoceros 3D software, using python . The user will find the design that appeals to her the most by exploring this evolutionary environment and evaluating the designs in each step until the satisfactory result is achieved.

The remainder of the thesis is organized as follows: in Section 2, I introduce the background and current methods used in fashion design as well as the concept of the applications of IGA. Section 3 gives the overview of the system. The system implementation and user interface are given in Section 4. The effectiveness of the pro-posed methodology is analyzed by experiments in Section 5. Finally, conclusions are summarized in Section 6.

BACKGROUND

Fashion Design Aid System

Today, advances in information technology along with the globalization of the world's economy have made a major change in the production process of many industries. The fashion industry, which is closely connected to our lives, has also shifted from small local production to mass production and then to make-to-order [1].

The fashion industry's main concern is to respond to clients' demands rapidly but at minimal cost [2, 3]. To achieve this goal, computer-aided fashion design can be used to enhance the efficiency of product development. Although there are different systems available for fashion design, such as Adobe Illustrator and AutoCAD, avail-able for fashion design, they are usually for professionals only and hard for non-professionals to use.

Appearance style is a metaphor for identity: through this kind of personal interpretation of or resistance to fashion, individuals announce who they are and who they hope to become [4]. Hence, a fashion design system that helps an ordinary person design her appealing fashion is very useful.

But what is "the most appealing fashion design"? People have different opinions on this matter, and the image they have in mind is often ambiguous and fuzzy. Therefore, it is difficult to design human evaluation functions. To reflect such human sensibility, this study proposes using Interactive Genetic Algorithm (IGA). IGA differs from the conventional Genetic Algorithms (GA) in its evaluation of the fitness function [5]. In conventional GA, the objective function is numerically defined, while in IGA, the fitness function is replaced by a human user. IGA "interacts" with users and incorporates the emotion and preference of the user in the process of evolution; therefore, the user's subjective evaluation determines the fitness of the solution [6]. IGA has been applied to optimization [7,8], designing the layout and lighting of rooms [9], fashion [10], web sites [11], data mining [12,13], and music composition [14], all according to the users' preferences.

Genetic Algorithm and Interactive Genetic Algorithm

Interactive genetic algorithms are a subset of genetic algorithms. Genetic algorithms, first explored by John Holland, are used to solve optimization and search problems by emulating principles of biological evolution [15]. GAs set variables up as genes and combine them to make designs. GAs then iteratively evaluate designs against a fitness test and select the best designs from a group to be used to make the next generation, these designs are subject to mutation and combination of traits, similar to biological evolution [16]. Instead of improving a single design, as seen in Fig. 1, a population of solutions is examined to find the best solutions, which are then recombined or mutated to generate a new population of (better) solutions. The general GA process is as follows [17]:

- Step 1: Initialize the population of chromosomes.
- Step 2: Calculate the fitness for each individual in the population using a fitness function.
- Step 3: Reproduce individuals to form a new population according to each individual's fitness.
- Step 4: Perform crossover and mutation on the population.
- Step 5: Go to step (2) until a particular condition is satisfied.

Here is the pseudocode [18]:

```
BEGIN
Create a random set of initial solutions
LOOP:
    Choose a subset of good solutions according to some ‘‘fitness measure’’.
    Perform recombination on randomly chosen pairs of solutions.
    Perform mutation on randomly chosen solutions.
UNTIL (population is stable)
END
```

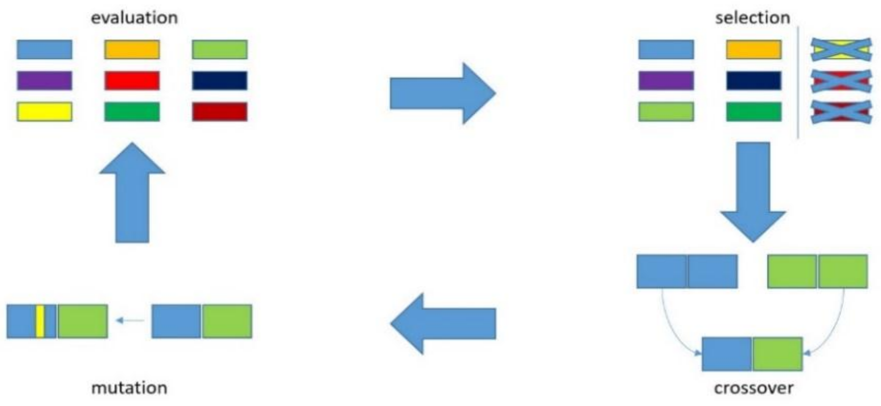
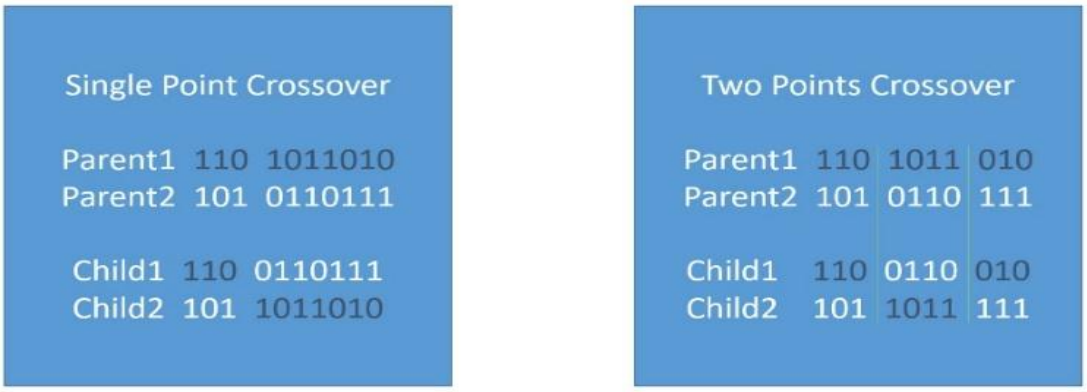


Fig.1. GA Process

To use GA for a problem, we should be able to parameterize the problem. Each parameter is considered a gene that can be represented through binary numbers. A combination of several genes creates a chromosome. Each singular representation of the chromosome within a population is termed a genotype [19]. An individual's fitness is calculated through decoding its genotype. In order to create new individuals for the next population, two individuals are submitted to crossover. To perform a single point crossover, as shown in Fig. 2, a crossover point is selected in each parents' genotype, then all data beyond that point is swapped between the two parents. The two-point crossover occurs by swapping everything between two selected points in parents' genotype. Fig.3 shows a visual representation of the crossover operation.



Single Point Crossover

Parent1 110 1011010
Parent2 101 0110111

Child1 110 0110111
Child2 101 1011010

Two Points Crossover

Parent1 110 1011 010
Parent2 101 0110 111

Child1 110 0110 010
Child2 101 1011 111

Fig.2. Left: Single Point Crossover. Right: Two Points Crossover

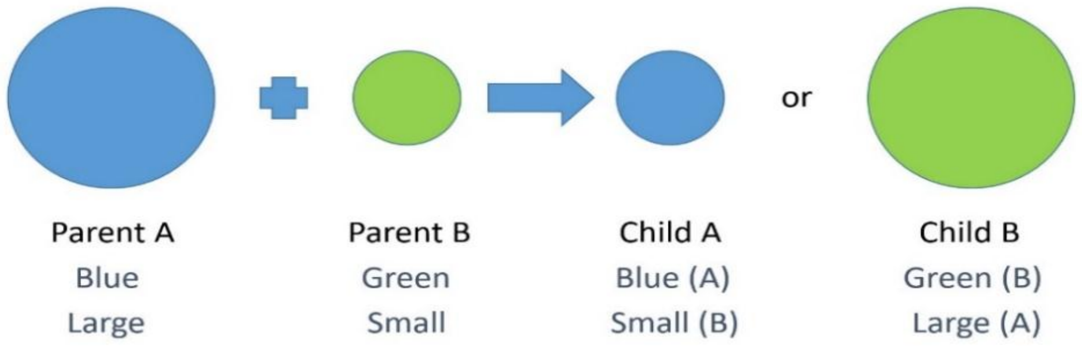


Fig.3. Visual Representation of Crossover Operation

To introduce more diversity, and potentially find drastically new solutions, mutation is utilized [19](Fig.4). Mutation occurs according to a user-definable mutation probability that is typically set to low. In the mutation process, as seen in Fig. 5, one point of the individual's genotype is modified.

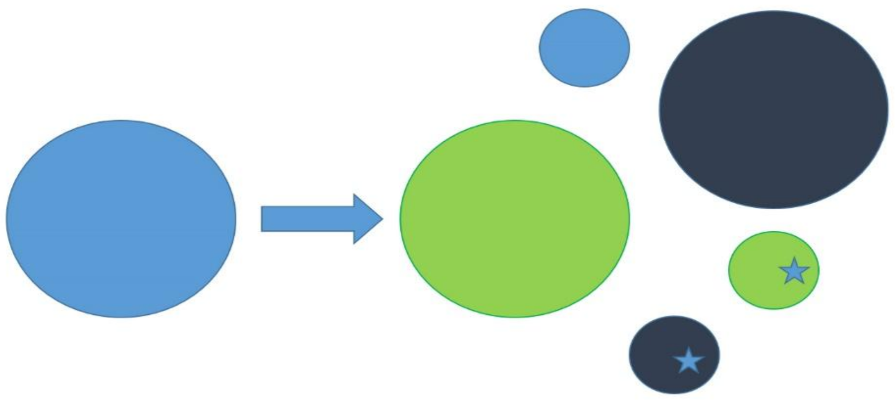


Fig.4. Visual Representation of Mutation Operation. In this example, those offspring marked with a star are those that required more than one mutation to produce.

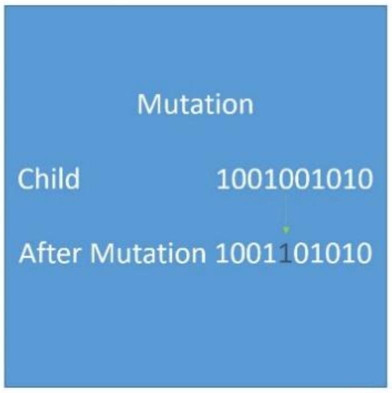


Fig.5. Mutation Operation

IGAs are similar to GAs, except the fitness function of the GA is replaced by a user evaluation of solution options. Human subject selects the most successful designs that will be used to create the next generation of designs. Interactive Genetic Algorithms were originally proposed by Dawkins [20] and led to works within computer art community. One example of evolutionary art is “Galapagos” exhibit by Karl Sims which allowed for discovery of interesting figures in a population of three-dimensional virtual organisms [21].

SYSTEM DESIGN

Types

GUD

A *GUD* (General Upper Design) is an integer in {1..10}. Each GUD refers to one of the 10 different definitions, designed with grasshopper for an upper body design piece. As an example, Fig. 6 shows the grasshopper definition for GUD number 9 and Fig.7 displays its representation in the rhinoceros interface. To be able to show this GUD, 3 initial values have been set for its list of parameters. Fig. 8 shows GUD number 9 with a different list of parameters. Parameters are discussed in more detail in the following sections.

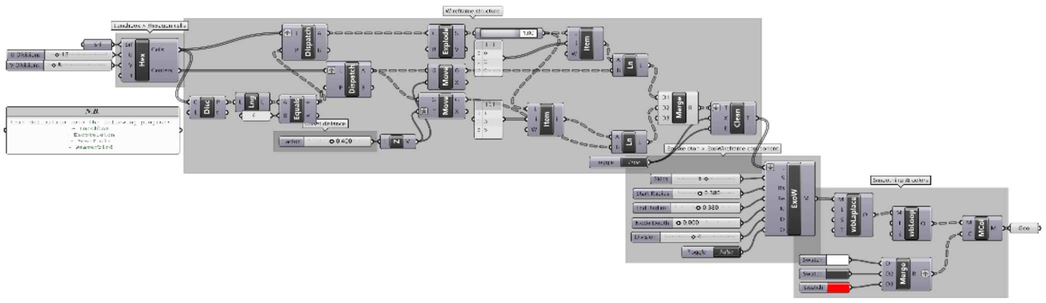


Fig.6. Grasshopper definition for GUD number 9

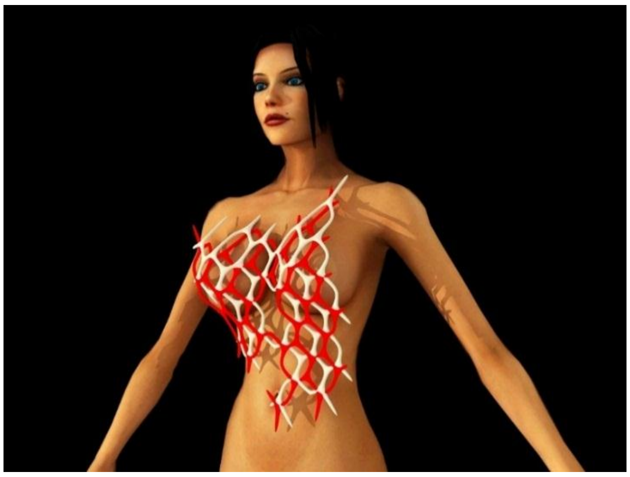


Fig.7. Representation of GUD number 9 in rhinoceros interface

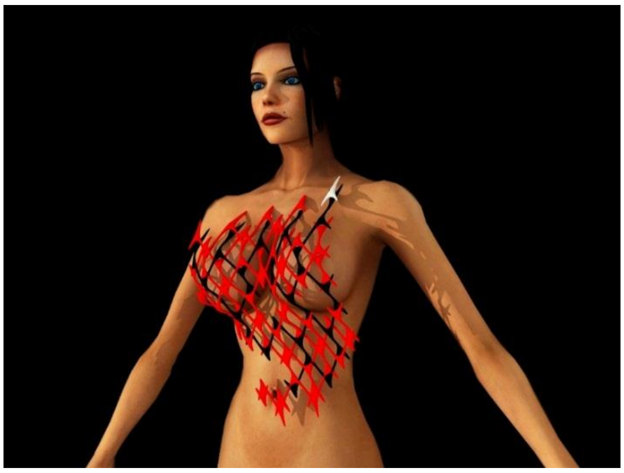


Fig.8. Representation of GUD number 9 in rhinoceros interface with a different list of parameters

GLD

A *GLD* (General Lower Design) is an integer in {1..10}. Each GLD refers to one of the 10 different algorithms, designed with grasshopper for a lower body design piece. As an example, Fig. 9 shows the grasshopper definition for GLD number 7 and Fig.10 displays its representation in the rhinoceros interface. To be able to show this GUD, 3 initial values have been set for its list of parameters. Fig. 11 shows GUD number 9 with a different list of parameters. Parameters are discussed in more detail in the fol-lowing sections.

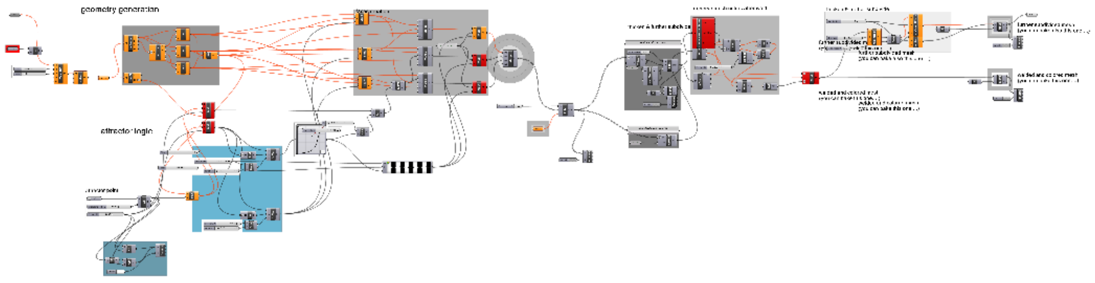


Fig.9. Grasshopper definition for GLD number 7

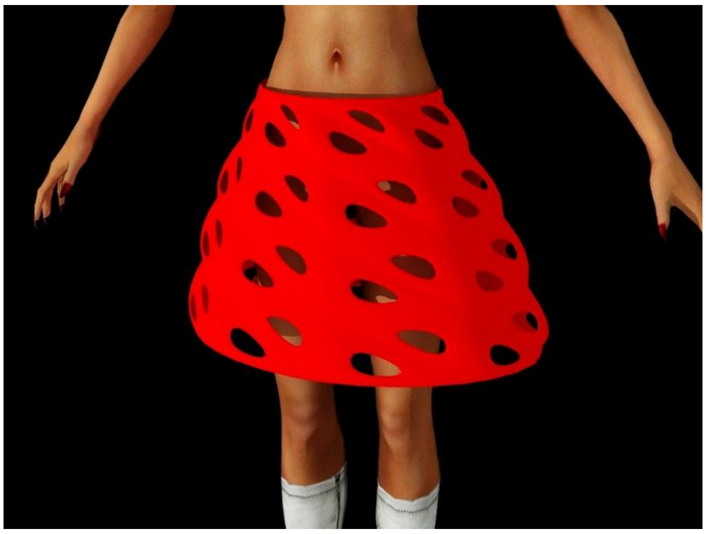


Fig.10. Representation of GLD number 7 in rhinoceros interface

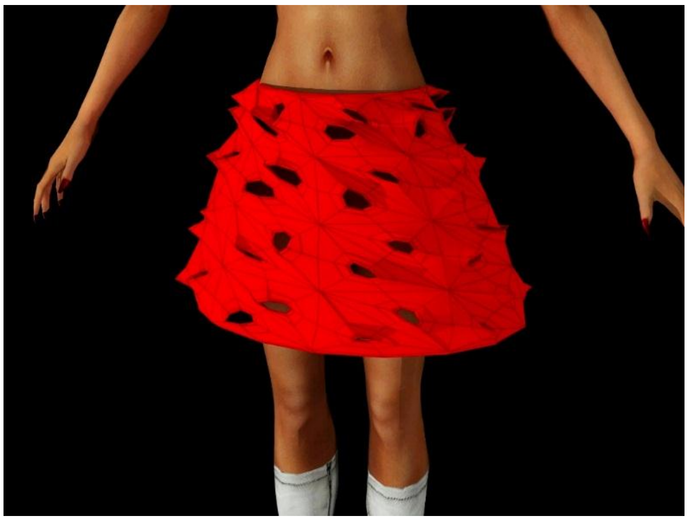


Fig.11. Representation of GLD number 7 in rhinoceros interface with a different list of parameters

Individual

An *individual* refers to a 2 piece dress consisting of Upper and Lower parts with spe-cific parameters. An individual is a dictionary of the form {'upper':X, 'lower':Y, 'up-perParms': L1, 'lowerParms':L2} where

- X is a GUD
- Y is a GLD
- L1 is a list of integers. Each of its members is a number in range(1,10).
- L2 is a list of numbers. Each of its members is a number in range(1,10).

As an example, Fig.12 shows an individual with the following dictionary: {‘upper’:9, ‘lower’:7, ‘upperParms’:[8,5.3,10],’lowerParms’:[5,9,0]}.

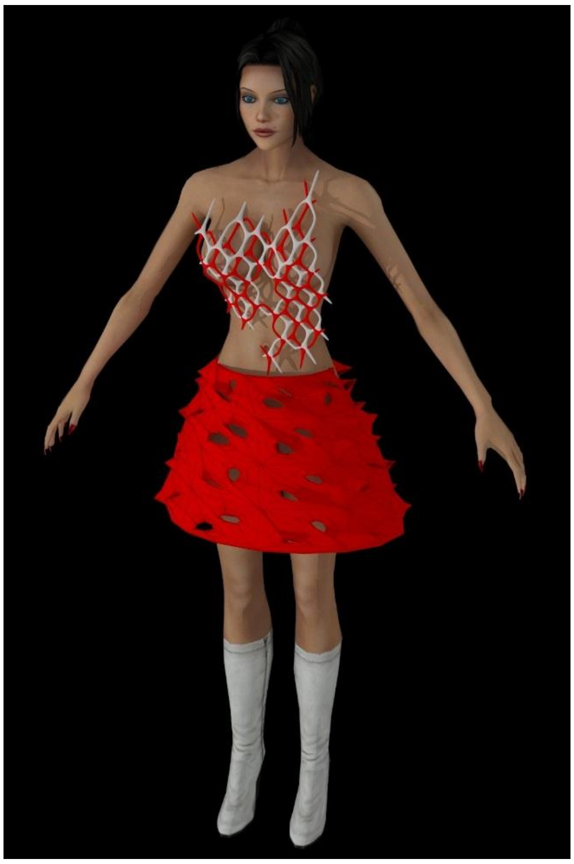


Fig.12. An individual with the following dictionary: {‘upper’:9, ‘lower’:7, ‘up-perParms’:[8,5.3,10],’lowerParms’:[4,10,5]}

Fig.13 shows an individual with the following dictionary: {‘upper’:9, ‘lower’:7, ‘up-perParms’:[8,5.3,10],’lowerParms’:[5,18,0]}. By comparing Fig.12 and Fig.13 we realize that a change in the parameters in ‘lowerParms’ will result in a change in GLD. In the case of ‘GLD’:7, the list of ‘lowerParms’ controls the size, shape and orienta-tion of the openings in the respective design.

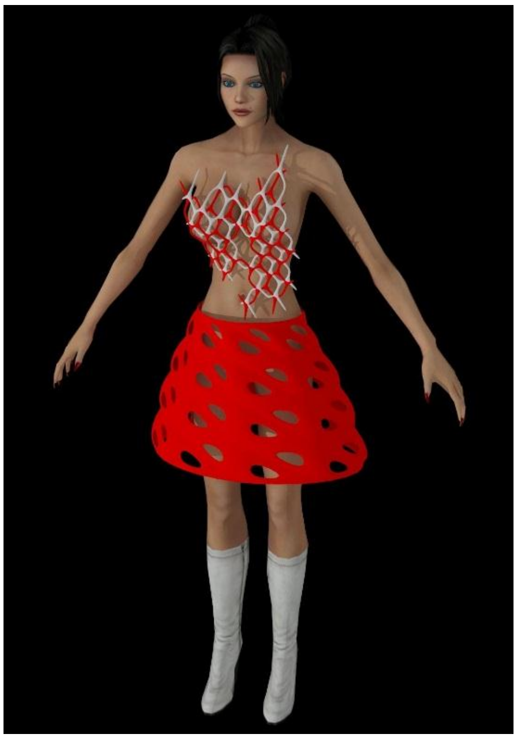


Fig.13. An individual with the following dictionary: {‘upper’:9, ‘lower’:7, ‘up-perParms’:[8,5.3,10],’lowerParms’:[5,18,0]}

Fig.14 shows an individual with the following dictionary: {'upper':9, 'lower':7, 'up-perParms':[10,6,5],'lowerParms':[5,9,0]}. By comparing Fig.13 and Fig.14 we realize that a change in the parameters in 'upperParms' will result in a change in GUD. In the case of 'GLD':7, the list of 'upperParms' controls the shape and number of vertical and horizontal divisions.

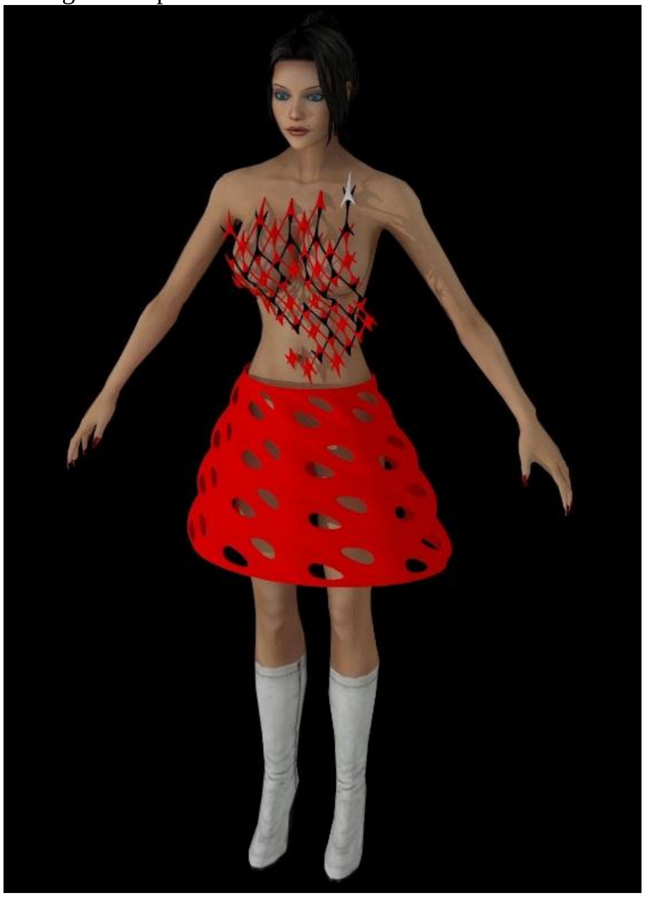


Fig.14. An individual with the following dictionary: {'upper':9, 'lower':7, 'up-perParms':[10,6,5],'lowerParms':[5,9,0]}

Population

A *population* is a list of 6 individuals.

Report

A *Report* is a list of 2 integers in the form of [X,Y], where X is an integer in {1..10} and Y is an integer in {1..10}. In a report[X,Y] , X is the user's favorite design in a population and Y is the user's second favorite design.

Functions

Crossover

Crossover is a random function from pairs of individuals to individuals. The new individual consists of either the upper part of the first and lower part of the second parent, or the upper part of the second and lower part of the first parent. If A is an individual with the dictionary: {'upper':9, 'lower':7, 'upperParms':[8,5.3,10], 'lowerParms': [5,9,0]} and B is an individual with the dictionary: {'upper':10, 'lower':5, 'upperParms':[3,3,10], 'lowerParms': [10,5,0.25]}, Fig 15. Shows possible individual C produced by their crossover with the following dictionary: {'upper':10, 'lower': 7, 'upperParms':[3,3,10],'lowerParms':[5,9,0]. This dictionary results in an individual with the upper design of individual B and lower design of individual A. Fig. 16 demonstrates individual D which is another possible individual produced by the crossover of A and B. Individual D has the upper design of individual A and lower design of individual B and is shown with the following dictionary: {'upper':9, 'low-er':5, upperParms': [3,3,10], 'lowerParms':[5,9,0]}.

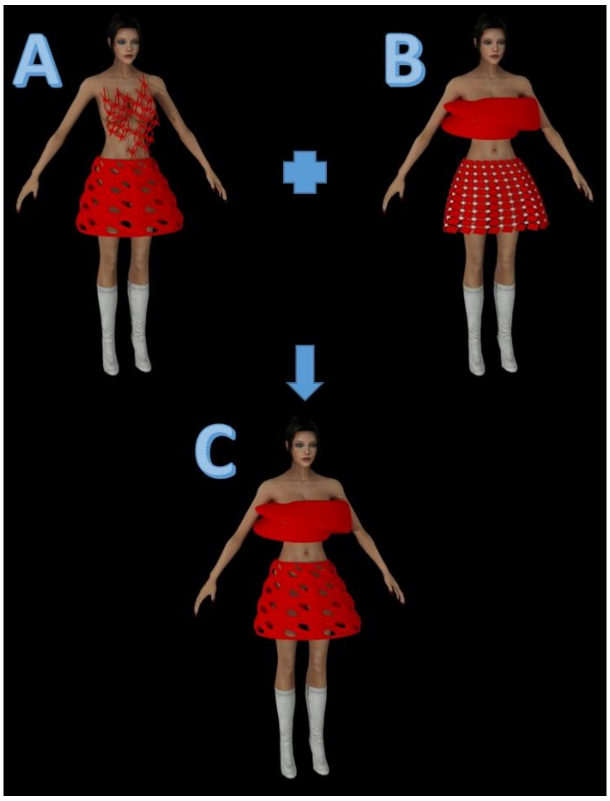


Fig.15. Crossover of Individual A with the dictionary {‘upper’:9, ‘lower’:7, ‘upperParms’:[8,5.3,10], ’lowerParms’: [5,9,0]} and B with the dictionary: {‘upper’:10, ‘lower’:5, ‘upperParms’:[3,3,10], ’lowerParms’: [10,5,0.25]} and their possible child C with the dictionary: {‘upper’:10, ‘lower’: 7, ‘upperParms’:[3,3,10],’lowerParms’:[5,9,0]

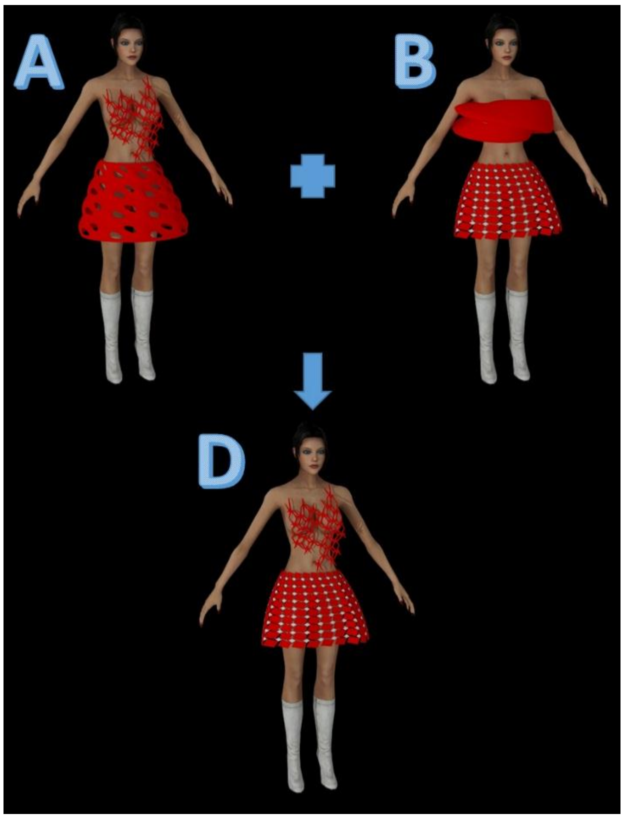


Fig.16. Cross over of Individual A with the dictionary {‘upper’:9, ‘lower’:7, ‘upperParms’:[8,5.3,10], ’lowerParms’: [5,9,0]} and B with the dictionary: {‘upper’:10, ‘lower’:5, ‘upperParms’:[3,3,10], ’lowerParms’: [10,5,0.25]} and their possible child D with the dictionary: {‘upper’:9, ‘lower’:5, upperParms’: [3,3,10], ’lowerParms’:[5,9,0]}

Mutation

Mutation is a random function from individuals to individuals, which changes a random element in either the list of upperParms or lowerParms of an individual. The process works as follows. The list of upperParm or lowerParms of an individual is chosen at random. An element from one of the 3 element in that list is chosen randomly and replaced with a randomly generated number in {0..10}

Fig.17 shows the individual we want to perform mutation on. This individual has the following dictionary: {'upper':3, 'lower':4, 'upperParms': [9,10,1], 'lower-Parms':[10,10,4]}.

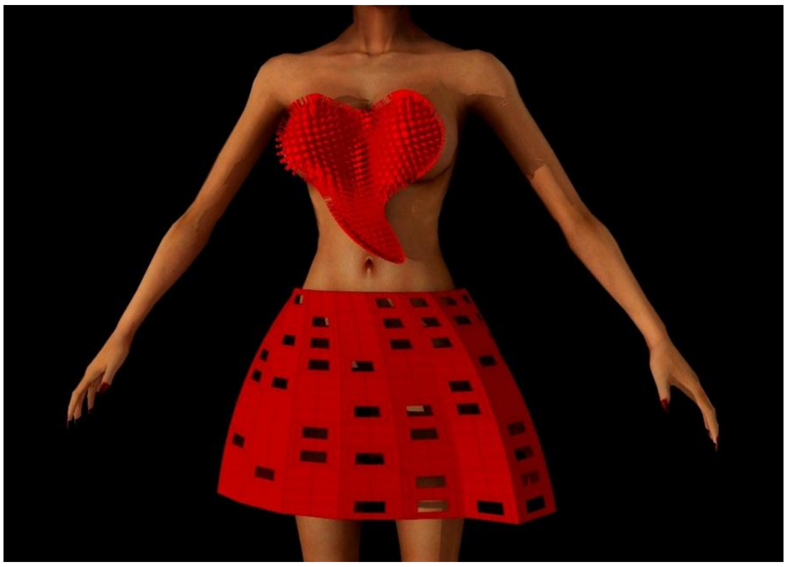


Fig.17. An individual with the dictionary: {'upper':3, 'lower':4, 'upperPArms: [9,10,1], 'lowerParms': [10,10,4]}

Performing mutation on Fig.17 can generate an individual with the following diction-ary: {'upper':3, 'lower':4, 'upperParms': [9,10,1], 'lowerParms':[10,10,6]} (Fig.18). The mutation operation has changed the third element in the list of lowerParms of that individual, causing a change in the placement of the openings in the lower design of that individual.

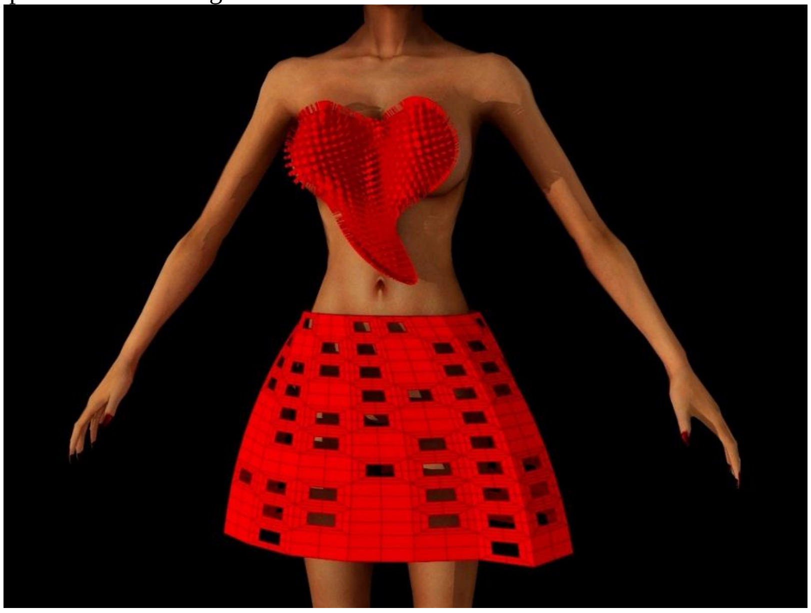


Fig.18. An individual, which is a mutation on the individual in Fig.17, with the dictionary: {'upper':3, 'lower':4, 'upperPArms: [9,10,1], 'lowerParms': [10,10,6]}

Performing mutation on Fig.17 can also generate an individual with the following dictionary: {'upper':3, 'lower':4, 'upperParms': [9,10,1], 'lowerParms':[10,5,6]} (Fig.19). This time, the mutation operation has changed the second element in the list of lowerParms of that individual, causing a change in the horizontal division of the lower design.

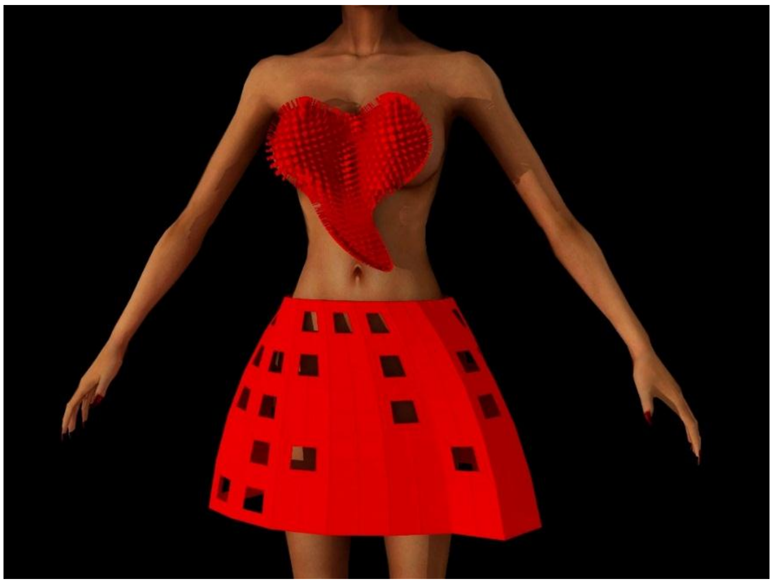


Fig.19. An individual, which is a mutation on the individual in Fig.17, with the dictionary: {‘upper’:3, ‘lower’:4, ‘upperPArms: [9,10,1], ‘lowerParms’: [10,5,4]}

Performing mutation on Fig.17 can also generate an individual with the following dictionary: {‘upper’:3, ‘lower’:4, ‘upperParms’: [6,10,1], ’lowerParms’:[10,10,4]} (Fig.20). This time, the mutation operation has changed the first element in the list of upperParms of that individual, causing a change in the horizontal placement of the spikes on the upper design.

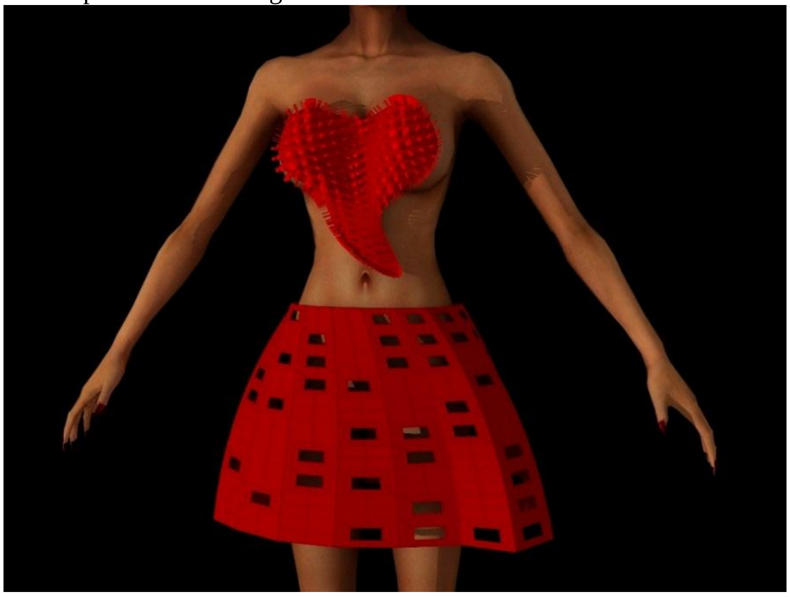


Fig.20. An individual with the dictionary: {‘upper’:3, ‘lower’:4, ‘upperPArms: [6,10,1], ‘lowerParms’: [10,10,4]}

Step

If x is the user's favorite dress in population p, y is the user's second favorite dress in population p, and a,b,c, and d are the remaining members of p, then step(p) is a new population consisting of the following individuals:

1. X
2. a mutation of x
3. a mutation of the crossover of x and y
4. a crossover of x with a random member of {a,b,c,d}
5. a crossover of mutation of a random member of {a,b,c,d} and a new individual
6. a new individual

The System Design

Fig. 21 shows the overview of the entire system. To create the initial population, the system selects random elements and combines them into 6 dress designs. User gives his fitness value by selecting 2 of his favorite designs in order of favorability. The system uses a pre-defined parent selection to choose pairs of designs for divergent and recombination. After applying crossover and mutation, the results are displayed again and this loop continues until a desired design is found. As a summary, here is a pseudocode description of the algorithm:

- step 1:Generation of the first population. The GUD, GLD, upperParms and lowerParms of each individual in this population are randomly selected.
- Step 2:display: The 6 individuals are presented to the user through the user inter-face.
- Step 3:Happy with the result? If so, end. If not, go to step 4.
- Step 4: Evaluation. In response to the presented population, the user is asked to give a report based on his subjective view. This evaluation will count as the fitness function in the IGA process.
- Step 5:reproduction. According to the choice of the user, either the convergent or divergent type of reproduction is used to create a new population. Go to step 2.

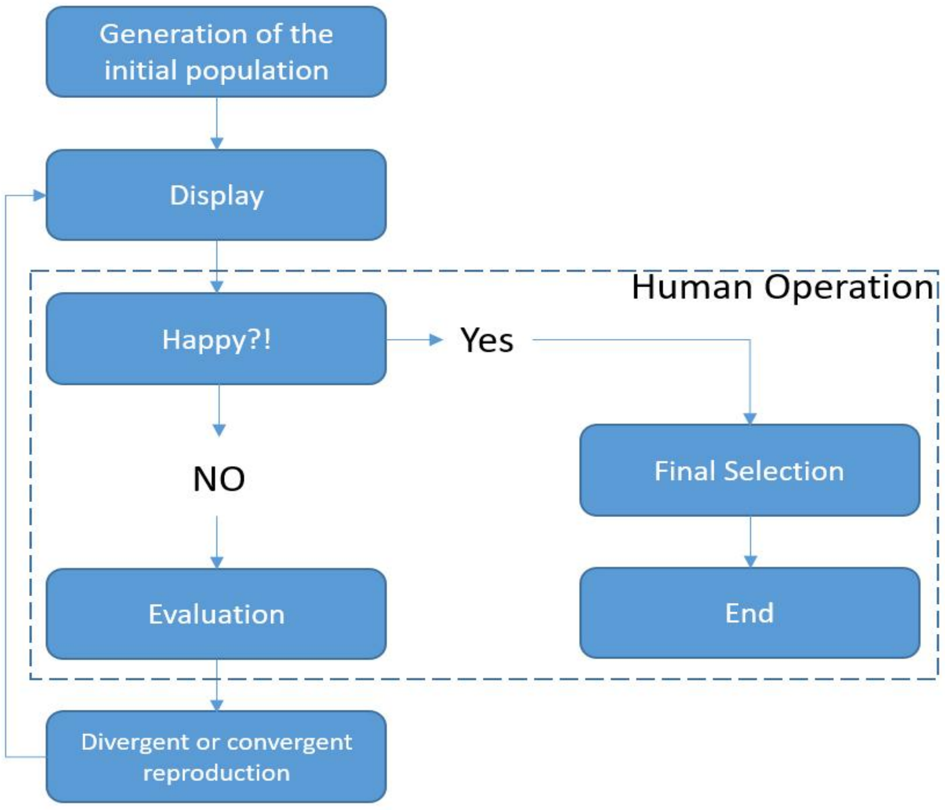

Generation of the initial population
Display
Human Operation
Happy?!
Yes
NO
Final Selection
Evaluation
End
Divergent or convergent reproduction

Fig.21. Flow diagram showing the system overview

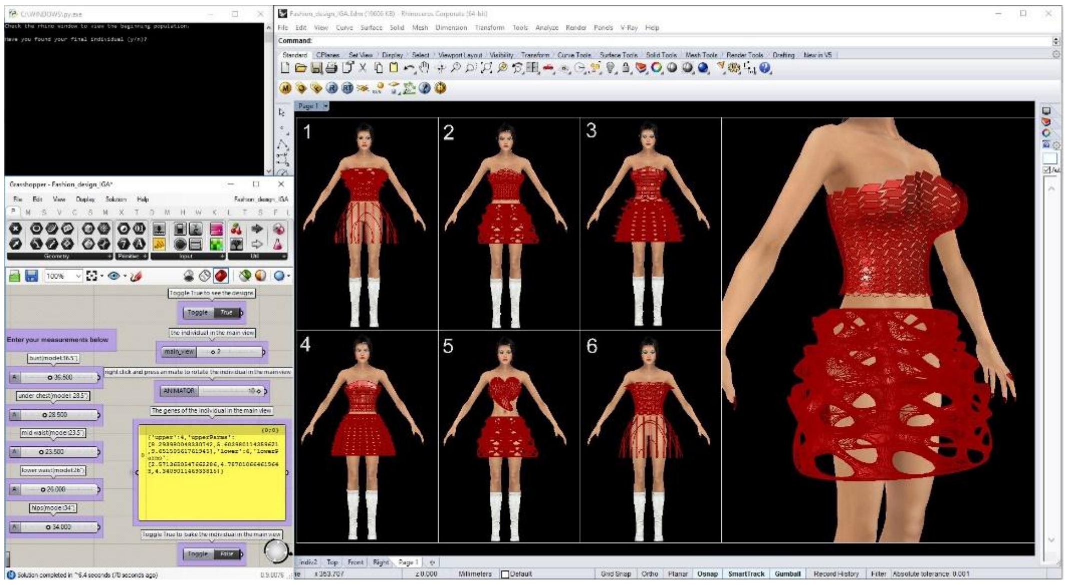
The overall system algorithm in python is provided in appendix A.

User's Fitness Value

Some researchers have proposed IGA systems in which users must rank each design within each set of a generation, while others have argued that such ranking causes user fatigue [19, 22, 23, 24]. Instead, they propose that users should identify their most and least preferred among a set because these are easier to identify than a whole ranking. This consequently reduces fatigue and provides a wealth of information [19]. For this research, 6 designs are displayed in each generation and the user will select his/her two favorite designs in each generation according to his preference. However, in the last generation the user will only select one final favorite design.

SYSTEM IMPLEMENTATION

The system is designed in the Rhinoceros and grasshopper software because of the precise designing tool they offer. Unfortunately, because of the current design decisions of the python component in grasshopper, the design of the system had to be transferred to Python shell, while the display of the population remained in Rhinoceros. Fig. 22 shows the user interface of the system (A demo of the interface can be found at https://www.youtube.com/watch?v=OJHHtnASRRU). The system displays current population composed of six designs (Fig. 23) in one screen. The number of the individual is shown next to each individual model. To decrease user fatigue, only 2 designs should be selected in each population. The selected designs have a higher probability to be used for recombination in the next generation.



```
Check the rhino window to view the beginning population.
Have you found your final individual (y/n)?
```

```
{'upper':4,'upperParms':
[0.293990049330742,5.603980114359621
,5.651505617619431],'lower':6,'lowerP
arms':
[2.5713620547662206,4.7870106646196
5,4.340901146333151]}
```

Solution completed in ~6.4 seconds (70 seconds ago)

C:\WINDOWS\py.exe
Grasshopper - Fashion_design_IGA*
Enter your measurements below
bust(model:36.5")
36.500
under chest(model: 28.5")
28.500
mid waist(model:23.5")
23.500
lower waist(model:26")
26.000
hips(model:34")
34.000
Toggle True to see the designs
Toggle True
the individual in the main view
main_view 2
right click and press animate to rotate the individual in the main view
ANIMATOR 10
The genes of the individual in the main view
Toggle True to bake the individual in the main view
Toggle False
1
2
3
4
5
6

Fig.22. User Interface

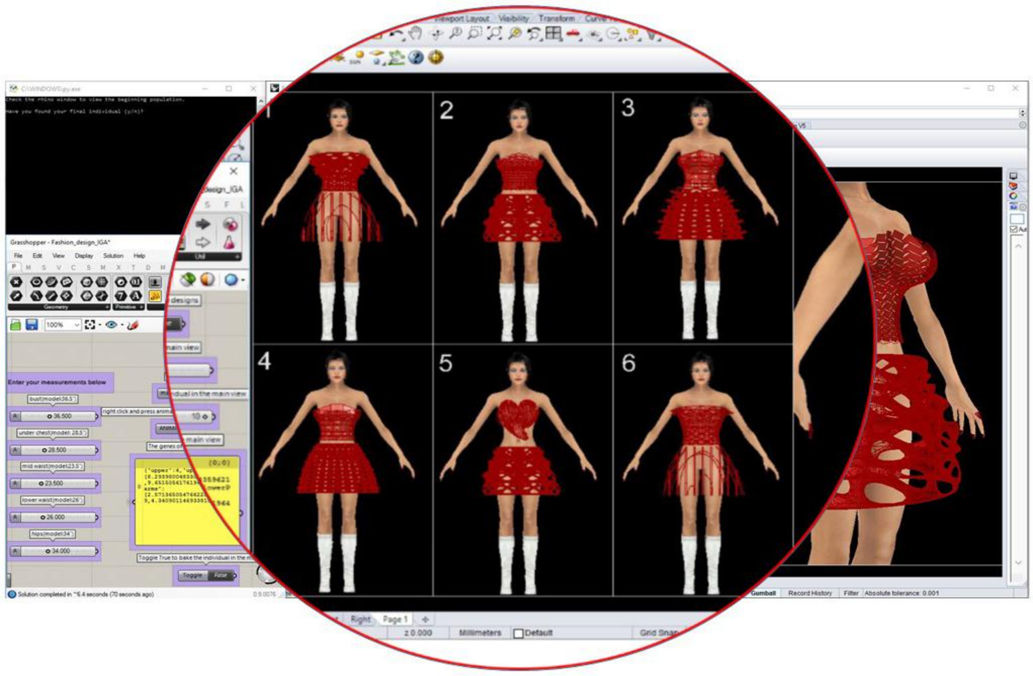
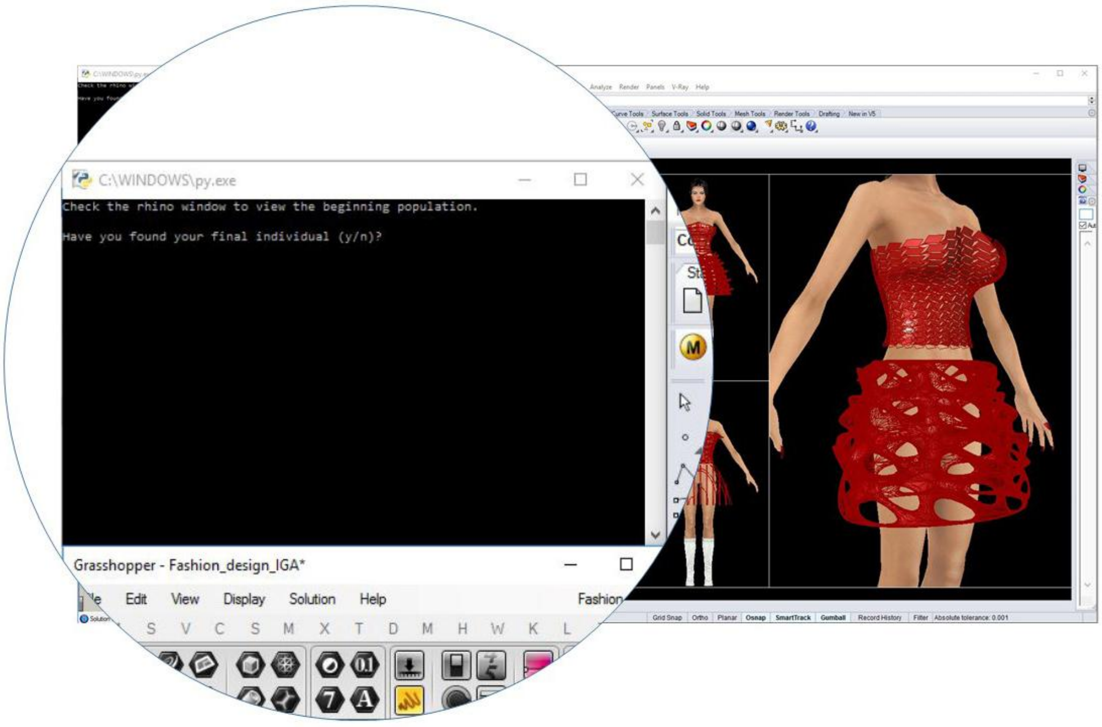


Fig.23. Population Example

As seen in Fig.24 upper left part of the screen shows the Python shell that lets the user interact with the system by providing the numbers of individuals for the report, stating if they want a new population to be generated, or if they want to end the loop.

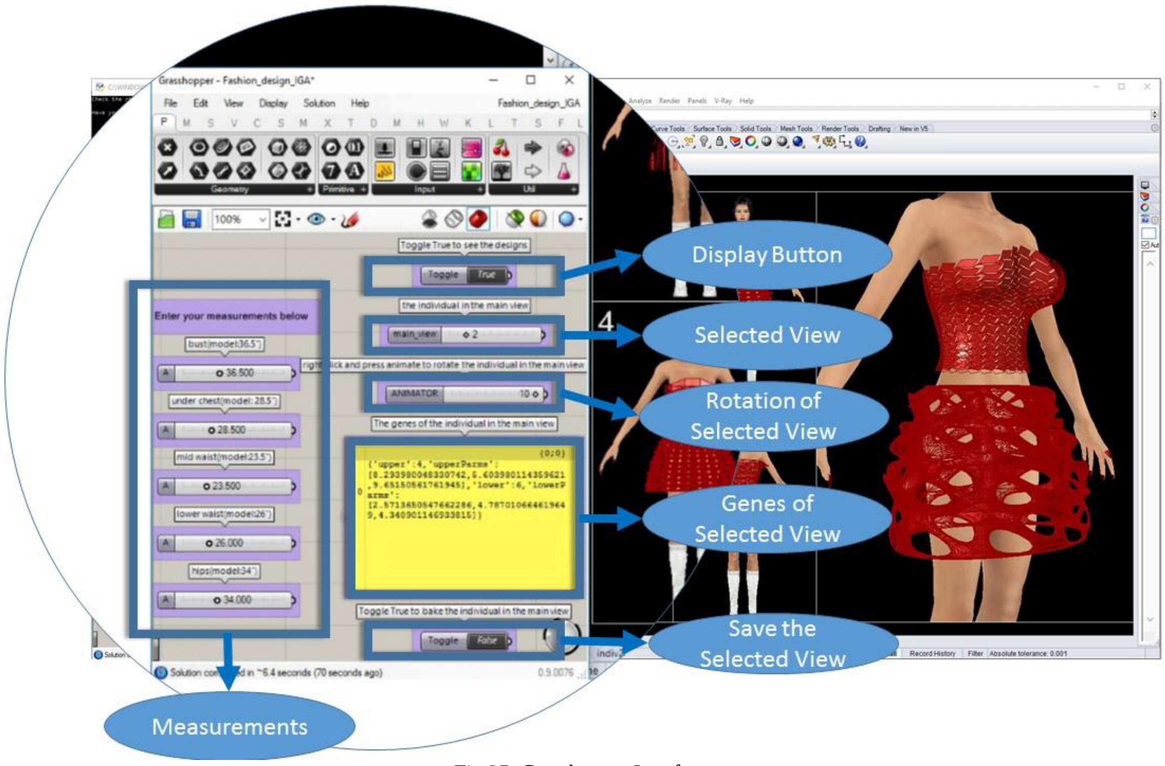

C:\WINDOWS\py.exe
Grasshopper - Fashion_design_IGA*
File Edit View Display Solution Help
Fashion

```
Check the rhino window to view the beginning population.

Have you found your final individual (y/n)?
```

Fig.24. Python shell user interface

User can input her measurements in the grasshopper interface (Fig.25). There is a button provided to display the population in the Rhinoceros interface.


Grasshopper - Fashion_design_IGA*
Enter your measurements below
bust(model:36.5")
36.500
under chest(model: 28.5")
28.500
mid waist(model:23.5")
23.500
lower waist(model:26")
26.000
hips(model:34")
34.000
Toggle True to see the designs
Toggle True
the individual in the main view
main_view 2
right click and press animate to rotate the individual in the main view
ANIMATOR 10
The genes of the individual in the main view
Toggle True to bake the individual in the main view
Toggle False
Display Button
Selected View
Rotation of Selected View
Genes of Selected View
Save the Selected View
Measurements

```
{0;0}
{'upper':4,'upperParms':
[8.293980048330742,5.603980114359621
,9.651505617619451],'lower':6,'lowerP
arms':
[2.571365054766228б,4.78701066461964
9,4.340901146933816]}
```

Fig.25. Grasshopper Interface

User can select a design to view in the right part of the screen and can rotate and zoom in the selected individual (Fig.26). To get more detail on a selected individual, its genes are shown in the grasshopper interface.

This design can be exported and saved for further use (e.g. 3D printing) with a click of a button provided in the grasshopper interface.

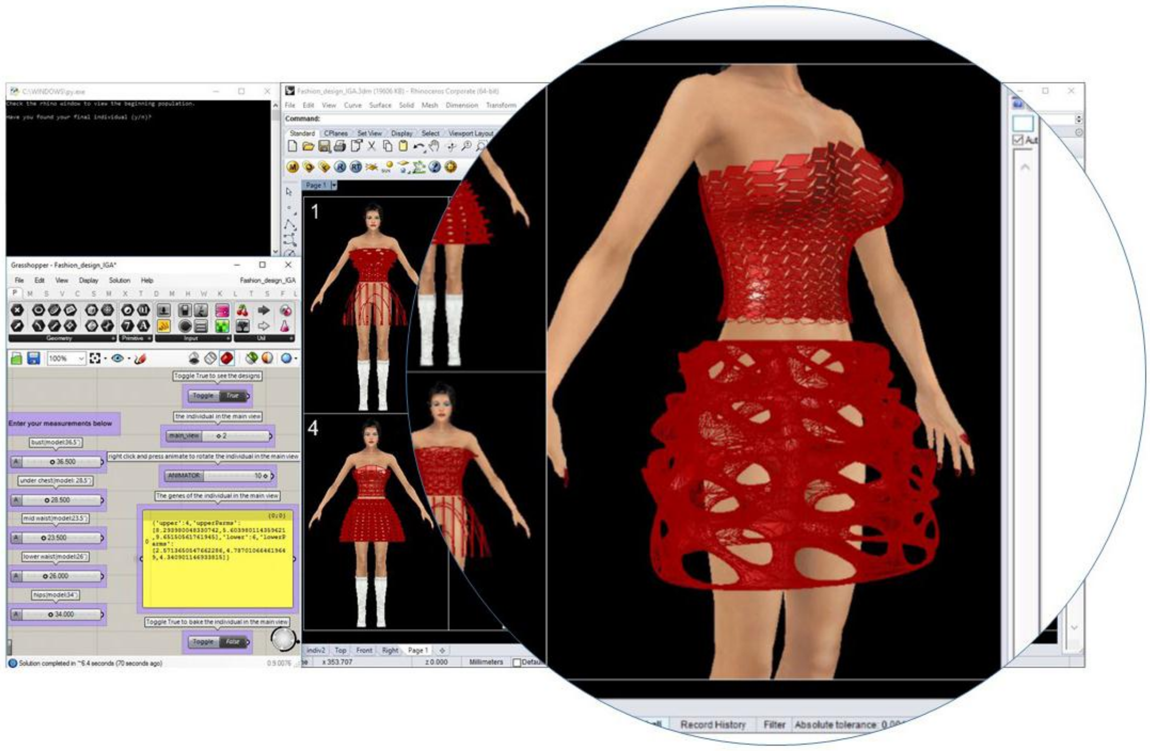


Fig.26. Selected view

IGAs can help enhance creativity because they can operate in both divergent and convergent ways [19]. We have implemented both operations in order to find new and interesting design concepts while retaining the characteristics that the user already liked. To model convergence, we gave a high probability to the designs selected by the user to become parents for the population of the new generation. Then, we applied a low mutation rate on the created designs. This process allows users to converge toward a design that is appealing to them.

Mistuned GAs may cause drastic diverge from their intended goal which increases in each generation. Thus, high mutation rates should be avoided in order to prevent random mutation during the process. In this system we managed to explore the design environment without diverging drastically from the main goal, while expecting unique inspiring designs.

EXPERIMENTAL OVERVIEW

Experimental Overview Number 1

To examine the performance of the system, we implemented it in both convergent and divergent process. In the convergent process, we allocated a lower mutation rate and a higher probability of the selected designs being used for creating the next population. For the divergent process, we allocated a higher mutation rate and a lower probability of the selected designs being used for the creation of the next population. This system allows for the implementation of both of these processes simultaneously. Our tests showed that the first 4-6 generations were considered divergent, and after that the generations tend to be more convergent. For This experiment, I modified the system, such that in each generation, the user would decide whether he wants the new population to have a higher randomness and mutation value (for exploring different designs) or a lower one that would let them converge toward a specific design of their choice. This experiment showed the improved system to be somewhat effective. (Fig.27) However, some users tend to use the divergent operator too often, which did not allow the system to be as convergent as it is intended to be. For further exploration of this system, we propose focusing on the limitation of the use of this operator to prevent this situation.

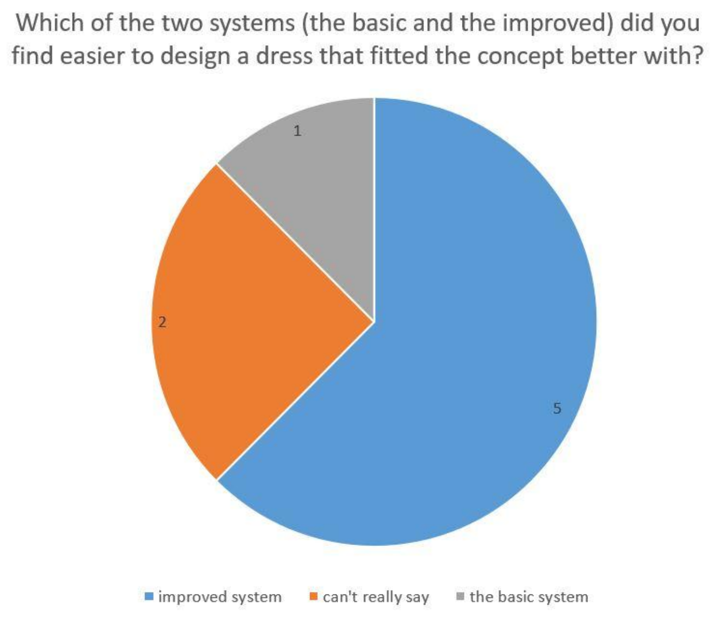


Fig.27. The result of the experimental overview number 1

Experimental Overview Number 2

We examined user satisfaction with our system by testing a group of 8 experiment subjects. They were asked to design an avant-garde dress that they would wear on a fashion runway using our proposed system and another online fashion design system. They were then asked to report which one of the systems they found more effective. Fig.28 shows the results of this experiment. The subjects were also asked to provide comments about both of these tools. They called the designs in the existing fashion tools mundane and not fashionable. However, they suggested an improvement on the proposed system to make it work faster. In addition, they asked for an easier interface, however, the interface was not an important part of this project, since the focus was mainly on the algorithms and system behind this program. They also asked for designs to be more realistic and suitable for daily use (instead of designs suitable for fashion runway).They stated that they would not necessarily prefer the proposed system over the online fashion aid system for a dress suitable for daily use. However, this was not the aesthetic measure that was decided to use in this project. We wanted the designs to be more structural and inspired by architecture, rather than normal designs for daily use. In conclusion, the experiments showed a promising result on the application of IGA for a fashion design aid system for non-professionals.

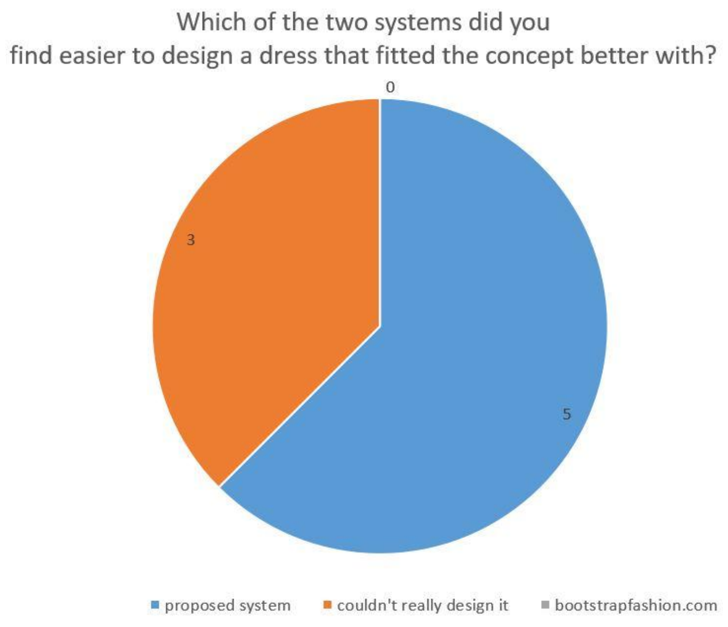


Fig.28. The result of the experimental overview number 2

Experimental Overview Number 3

We examined user satisfaction with the proposed IGA system in contrast to the basic GA system by testing a group of 8 experiment subjects. We asked the subjects to design a cool-looking costume, using the proposed IGA system. They were asked to do the same thing also by using the basic GA system, which does not ask for the user input to create each population. Instead, in each generation, the basic GA system generates the new population by random. They had to find their favorite design in less than 10 generations. The results showed the IGA system to be more effective than basic GA system.

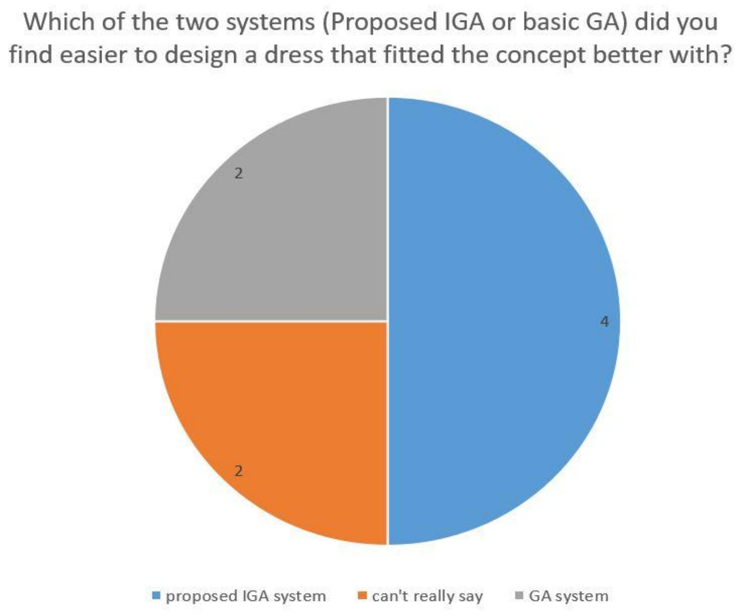


Fig.29. The result of the experimental overview number 3

Experimental Overview Number 4

It is difficult to show the convergence of IGA with quantitative analysis because it is operated based on human's evaluation. We have requested 5 subjects to find cool looking and elegant design using the system for 10 generations. They were asked to rank their favorite design in each generation from 1 to 5. Fig.30 shows the changes of the grade on average. This figure shows the steady improvement of the result over generations. The cool looking case shows better results than the elegant one, because the meaning of 'elegant' might be more complex and various than that of 'cool-looking, or it might be that the proposed system can produce dresses with cool looking value rather than elegance.

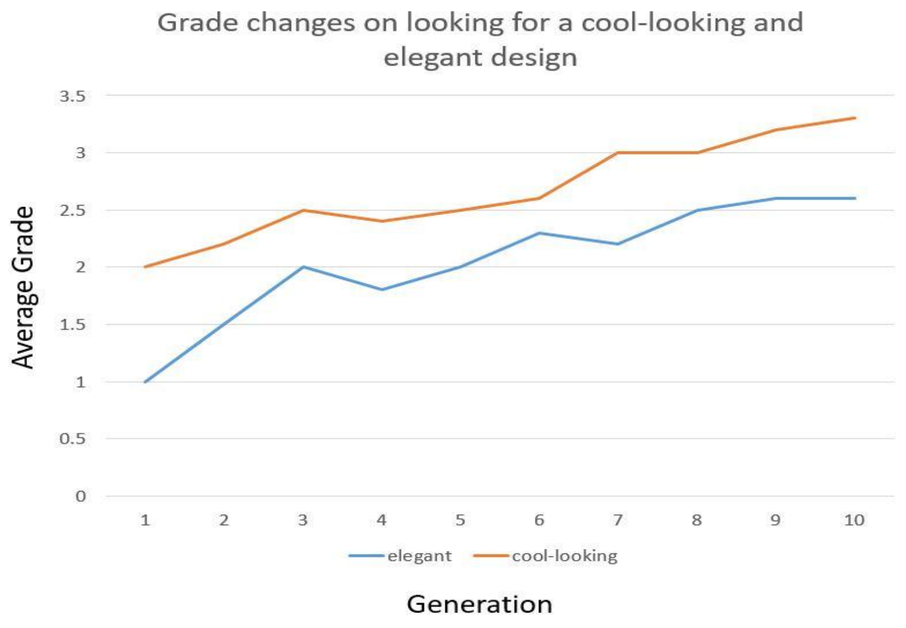


Fig.30. The result of the experimental overview number 4

Future Development

An important aspect of human computer interaction is the idea that a computer system should allow the user to make use of their increased knowledge about the system as they use it more frequently [25]. As users gain more experience with the system, they grow some understanding of how different parameters affect the design. To use this knowledge, we propose an option in the interface to adjust values of those parameters, which affect not the absolute values of those parameters but how much the algorithm explores those regions of parameter space, i.e. the extent to which corresponding regions of the genome are mutated [18]. In addition, an option to adjust the exploration rate could be also useful. This option could provide the user with control over the mutation and crossover rate.

CONCLUSION

This study proposed a fashion design aid system for non-professionals by using IGA. The proposed system, which is designed in the Rhinoceros and grasshopper software, with utilization of python shell uses user's taste as a fitness value to create a large number of design options. The experimental tests showed that users are significantly satisfied with the system; therefore using IGA for a fashion design aid system is a proper choice.

We tested the performance of our system in both convergent and divergent ways. The results showed the system to be effective in both ways. These two concepts can help enhance creativity. With an appropriate implementation of both these operations, we can allow for reaching to an appealing design, while exploring the unique interesting designs in our system environment.

To improve our system, we propose to add an option in the interface to adjust the parameters that affect the design. In addition, an option to adjust the exploration rate could be also useful. This option can provide the user with control over the mutation and crossover rate.

We also suggest adding a mutation operator allowing local exploration of a selected design. To further improve this study, we propose to focus on limitation of this operator, to prevent the system from being one-dimensional and not being able to explore the design environment.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Flueckiger for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge.

Besides my advisor, I would like to thank the rest of my thesis committee. My sincere thanks goes to Dr. Rushton who provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general.

Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models. I am also grateful to my other family members and friends who have supported me along the way.

BIBLIOGRAPHY

[1] M'Hallah, R., Bouziri, A.: Heuristics for the combined cut order planning two-dimensional layout problem in the apparel industry. Int. Trans. Oper. Res. 23(1–2), 321–353 (2016)

[2] Guo, Z.X., Wong, W.K., Leung, S.Y.S., Fan, J.T., Chan, S.F.: Mathematical model and genetic optimization for the job shop scheduling problem in a mixed and multi-product assembly environment: a case study based on the apparel industry. Comput. Ind. Eng. 50, 202–219 (2006)

[3] Rose, D., Shier, D.: Cut scheduling in the apparel industry. Comput. Oper. Res. 24, 3209– 3228 (2007)

[4] Kaiser, S.B.: Fashion and Cultural Studies. Berg, London (2012). English edition

[5] Gonsalves, T., Kawai A.: Fourth International conference on Computer Science & Information Technology, pp. 169–174 (2014)

[6] Hu, Z.-H., Ding, Y.-S., Zhang, W.-B., Yan, Q.: An interactive co-evolutionary CAD system for garment pattern design. Comput. Aided Des. 40(12), 1094–1104 (2008). doi:http://dx.doi.org/10.1016/j.cad.2008.10.010

[7] Sakawa, M., Yauchi, K.: Interactive decision making for multiobjective nonconvex programming problems with fuzzy numbers through coevolutionary genetic algorithms. Fuzzy Sets Syst. 114(1), 151–165 (2000)

[8] Sakawa, M., Nishizaki, I.: Interactive fuzzy programming for two-level nonconvex programming problems with fuzzy parameters through genetic algorithms. Fuzzy Sets Syst. 127(2), 185–197 (2002)

[9] Fukada, Y., Sato, K., Mitsukura, Y., Fukumi, M.: The room design system of individual preference with IGA. In: International Conference on Control, Automation and Systems, Seoul, Korea (2007)

[10] Kim, H.S., Cho, S.B.: Application of interactive genetic algorithm to fashion design. Eng. Appl. Artif. Intell. 13(6), 635–644 (2000)

[11] Oliver, A., Monmarche, N., Venturini, G.: Interactive design of web sites with a genetic algorithm. In: Proceedings of the IADIS International Conference WWW/Internet, Lisbon, Portugal, pp. 355–362 (2002)

[12] Kim, H.-S., Cho, S.-B.: Application of interactive genetic algorithm to fashion design. Eng. Appl. Artif. Intell. 13(6), 635–644 (2000)

[13] Gong, D.-W., Hao, G.-S., Zhou, Y., Sun, X.-Y.: Interactive genetic algorithms with multipopulation adaptive hierarchy and their application in fashion design. Appl. Math. Comput. 185(2), 1098–1108 (2007)

[14] Tokui, N., Iba, H.: Music composition with interactive evolutionary computation. In: Proceedings of the Generative Art 2000, International Conference on generative Art, Milan, Italy (2000)

[15] Holland, J.: Adaptation in Natural and Artificial System. The University of Michigan Press, Ann Arbor (1975)

[16] Nathan-Roberts, D.: Using Interactive Genetic Algorithms to Support Aesthetic Ergonomic Design. Dissertation, University of Michigan. Ann Arbor, Michigan: ProQuest/UMI (inpress)

[17] Eberhart, R., Simpson, P., Dobbins, R.: Computational Intelligence PC Tools. Waite Group Press, Corte Madera (1996)

[18] Johnson, C.B.: Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In: Patrizio, A., Wiggins, G.A., Pain, H. (eds.) Proceedings of the AISB 1999 Symposium on Artificial Intelligence and Musical Creativity. Brighton: Society for the Study of Artificial Intelligence and Simulation of Behaviour (1999)

[19] Kelly, J.C.: Interactive genetic algorithms for shape preference assessment in engineering design. ProQuest (2008)

[20] Dawkins, R.: The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design. Norton, New York (1996)

[21] Frauenfelder, M.: Do-it-yourself darwin. Wired 6(10), 164 (1998)

[22] Buonanno, M.A., Mavris, D.N.: Small supersonic transport concept evaluation using interactive evolutionary algorithms. In: Collection of Technical Papers – AIAA 4th Aviation Technology, Integration, and Operations Forum, ATIO, vol. 1, pp. 411–426, 20–23 September 2004

[23] Cho, S.B.: Towards creative evolutionary systems with interactive genetic algorithm. Appl. Intell.: Int. J. Artif. Intell. Neural Netw. Complex Probl. Solving Technol. 16(2), 129–38 (2002)
[24] Kamalian, R., Zhang, Y., Takagi, H., Agogino, A.M.: Reduced human fatigue interactive evolutionary computation for micromachine design. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 9 (2005)
[25] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T.: Human-Computer Interaction. AddisonWesley, Essex (1994)

APPENDIX A
THE SYSTEM ALGORITHM

```
import random
#import time
a=[]
#random.seed(time.clock())
f = open('thesis.txt', 'r+')
#GUD.
#A GUD (General Upper Design) is an integer in {1..10}.
#Each GUD refers to one of the 10 different definitions, designed with grasshopper for an upper body design piece.

#GLD.
#A GLD (General Lower Design) is an integer in {1..10}.
#Each GLD refers to one of the 10 different algorithms, designed with grasshopper for a lower body design piece.

#Individual.
#An individual refers to a 2 piece dress consisting of Upper and Lower parts with spe-cific parameters. An individual is a dictionary of the form {'upper':X, 'lower':Y, 'up-perParms': L1, 'lowerParms':L2} where
#X is a GUD
#Y is a GLD
#L1 is a list of integers. Each of its members is a number in range(0.1,10).
#L2 is a list of numbers. Each of its members is a number in range(0.1,10).

#Population.
#A population is a list of 6 individuals.

#Report.
#A Report is a 2-tuple of integers in the form of (X,Y), where X is an integer in {1..10} and Y is an integer in {1..10}.
#In a report(X,Y) , X is the user's favorite design in a population and Y is the user's second favorite design.

#Crossover.
#Crossover is a random function from pairs of individuals to individuals.
#The new individual consists of either the upper part of the first and lower part of the second parent, or the upper part of the second and lower part of the first parent.
#Crossover: individual*individual->individual
def crossover(individual1,individual2):
   X1= {'upper':individual1['upper'], 'lower': individual2['lower'], 'upperParms': individual1['upperParms'], 'lowerParms': individual2['lowerParms'] }
   X2= {'upper': individual2['upper'], 'lower': individual1['lower'], 'upperParms': individual2['upperParms'], 'lowerParms': individual1['lowerParms'] }
   return random.choice([ X1,X2])

#Mutation is a random function from individuals to individuals, which changes a random element in either the list of upperParms or lowerParms of an individual.
#The process works as follows. The list of upperParm or lowerParms of an individual is chosen at random.
#An element from one of the 3 element in that list is chosen ran-domly and replaced with a randomly generated number in {0..10}
#mutation: individual->individual
def mutation(individual):

A=[individual['upperParms'][0],individual['upperParms'][1],individual['upperParms'][2]]

B=[individual['lowerParms'][0],individual['lowerParms'][1], individual['lowerParms'][2]]
   X1= random.randint(0,1)
   X2= random.randint(0,2)
   if X1==0:
      A[X2]=random.uniform(0.1,10)
   else:
      B[X2]=random.uniform(0.1,10)
   return {'upper': individual ['upper'], 'lower':individual['lower'], 'upperParms': A, 'lowerParms':B}


#initPopulation() is a random function that produces a list of 6 individuals.
#initPopulation:list of individuals
def initPopulation():
   individuals=[]

a={'upper':random.randint(1,10),'lower':random.randint(1,10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)]}

b={'upper':random.randint(1,10),'lower':random.randint(1,10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)]}

c={'upper':random.randint(1,10),'lower':random.randint(1,10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.1,10),random.uniform(0.1,10)]}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

```
d={'upper':random.randint(1,10),'lower':random.randint(1,
10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)]}

e={'upper':random.randint(1,10),'lower':random.randint(1,
10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)]}

f={'upper':random.randint(1,10),'lower':random.randint(1,
10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)]}
    individuals.extend([a,b,c,d,e,f])
    return individuals

#Step.
#step is a function from population*report which generates
a new population according to the beginning population and
report(x,y),
#where report consists of the user's favorite designs in that
beginning population.
#step:Population*report->Population
def step(population,report):
    individuals=[]
    y=population
    removes=[report[0],report[1]]
    y=[i for j, i in enumerate(y) if j not in removes]
    #y is now the list of all other individuals in the beginning
population except the favorites.
    a=population[report[0]-1]
    b=mutation(population[report[0]-1])
    c=mutation(crossover(population[report[0]-
1],population[report[1]-1]))

d=mutation(crossover(random.choice([population[report[0
]-1],population[report[1]-1]]),random.choice(y)))

e=crossover(mutation(random.choice(y)),{'upper':random.
randint(1,10),'lower':random.randint(1,10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)]})

f={'upper':random.randint(1,10),'lower':random.randint(1,
10),
'upperParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)],
'lowerParms':[random.uniform(0.1,10),random.uniform(0.
1,10),random.uniform(0.1,10)]}
    individuals.extend([a,b,c,d,e,f])
    return individuals

##################################################
##########
population=initPopulation()
print("Check the rhino window to view the beginning
population.")
f.write(str(population))
f.close()
while True:
    x=input("\nHave you found your final individual (y/n)?")
    if x=='y':
        print("\nIn the grasshopper window choose your final
design to be shown in the main view,then bake it and
enjoy!\n")
        ex=input("Type 'y' when you are ready to exit?")
        if x=='y':
            break;
    elif x=='n':
        print("\nTo continue you need to select 2 of your
favorie designs.\n")
        report=[0,0]
        report[0]=int(input("The number of your first favorite
design?"))
        report[1]=int(input("\nThe number of your second
favorite design?"))
        population=step(population,report)
        f = open('thesis.txt', 'r+')
        f.write(str(population))
        f.close()
        print("\nCheck the rhino window to view the new
population.")
```