A Genetic Algorithm Approach Towards Image Optimization

Mujahid Tabassum, Kuruvilla Mathew and Sailesh Choytooa
Swinburne University of Technology, Sarawak Campus, Jalan Simpang Tiga, 93350, Kuching, Sarawak, Malaysia
mtabassum@swinburne.edu.my; kmathew@swinburne.edu.my; 4226623@students.swinburne.edu.my

ABSTRACT

In today's world, an optimal and intelligent problem solving approaches are required in every field, regardless of simple or complex problems. Researches and developers are trying to make machines and software's more efficient and intelligent. This is where the Artificial Intelligence plays its role in developing efficient and optimal searching algorithm solutions. Genetic algorithm is one of most pervasive and advanced developed heuristic search technique in AI. Genetic algorithm (GA) is developed to find the most optimized solution for a given problem based on inheritance, mutation, selection and some other techniques. GA has proved itself to be a powerful, unbiased optimization technique in today's industry. It was proved that genetic algorithms are the most powerful unbiased optimization techniques for sampling a large solution space. They were applied for the image enhancement, segmentation, feature extraction and classification as well as the image. In this paper, we have included the genetic algorithm flowchart with basic parameters that include the cross-over probability, number of cross-over points, mutation probability, maximum number of iterations and population size. A case study on how images can be reproduced using the optimal parameters is conducted. The image used was the SUTS logo and it was reproduced using the GA.

KEYWORDS

Artificial Intelligence, Genetic Algorithms; Optimization; GA operators; Metaheuristics; JNetic

1 INTRODUCTION

A genetic algorithm is an artificial intelligence search metaheuristic that is derived from the process of biological organism evolution. Genetic algorithms were described in University of Michigan 1960s and 1970s by John Holland [1].

Genetic algorithms are a subset of the evolutionary algorithm set. The design of evolutionary algorithms is based on processes which occur in living organisms in nature, for example, inheritance, mutation, selection and crossover.

GA's are uniquely distinguished by having a parallel-population based search with stochastic selection of many individual solutions, stochastic crossover and mutation. Many other search methods have some of these elements, but only GA's have this particular combination [2]. Genetic algorithms are used various fields of biology, biotechnology, computer science, engineering, economics, chemistry, manufacturing, mathematics, medicine and pharmacology and other fields and have numerous advantages and disadvantages.

In the event the reader finds difficulty understanding biological concepts, an appendix relating to the explanation of those concepts has been included. However, the reader is expected to be familiar with main concepts and keywords used in the field of artificial intelligence.

In this paper a real world problem has been included as a case study to evaluate the application and performance of genetic algorithm. It deals with image reproduction given a sample image, the GA must reproduce this image using various parameters.

The purpose of this case study is to find the parametric values of the GA that provides the most optimum solution (maximum generation, optimum population, optimum crossover rate, optimum mutation rate).

2 RELATED WORKS

Genetic algorithm have quite a number of advantages which makes it as one of the most preferable and widespread search algorithms in Artificial Intelligence. One of the advantages is the capability of solving any optimization problem based on chromosome approach, another remarkably important feature of this algorithm

is its capability to handle multiple solution search spaces and solve the given problem in such an environment. Moreover, genetic algorithms are less complex and more straightforward compared to other algorithms. In addition Genetic algorithms are easier to be transferred and applied in different platforms, thereby increasing its flexibility. Many algorithms in AI are complete and optimal in finding a solution; however, these algorithms are only efficient for single objective solutions which mean there is only a single criterion for the solution state. Whereas, looking for solutions with multiple objectives are much more complex. In such an environment each solution will have more than one criteria and every change in one of these criteria will directly affect the others. Genetic algorithm has this capability to generate efficient solutions in such a complex and mathematically sophisticated environments. This process is performed by using the described implementation operators such as crossover and mutation where the new solutions will inherit from more than one previous chromosome. As a result, as the search space is expanded down the hierarchy each chromosome will be actually the outcome of merging different genes of the previous most optimal solutions. This will subsequently result in finding the final optimal solution which is derived from merging all of the previous generations' optimal solutions (at their own generation time) which is the final desirable outcome in a multiple objective environment [6].

2.1 Genetic Algorithm Applications

The flexibility of genetic algorithm has resulted in manipulation of this field in various applications over different industries. Some of these applications are discussed as follows [11]:

2.1.1 Evolvable hardware applications

This field is based on manipulation of GA to produce electronic. The Genetic algorithm models exploit stochastic operators to automatically derive new configurations based on the old configurations. As the model keeps evolving while running in its environment context, finally the desirable configuration which is required by the designer will be reached. An example for such a reconfigurable model can be a robot with capability of manipulating built-in GA to regenerate its configuration after some breakdown due to environmental issues such electromagnetic wave which can cause malfunction in its normal configuration. Moreover, such a robot will be able to alter its configuration to a newer version if it meets a situation where it is requires more functionalities to perform its tasks.

2.1.2 Robotics

As it is clear robotics necessitate designers and engineers to experiment and figure out all the required aspects of a robot such as hardware infrastructure and corresponding software architecture to develop a comprehensive and efficient robot. For any new task all the mentioned activities are required to be performed again to design a new robot which suits the new objectives. By manipulating Genetic algorithm many of these extra designs requirements can be eliminated. Genetic algorithm will provide this capability to automatically generate a collection of optimal designs which can be used for specific tasks and activities. This approach can even be expanded and result in the generation of robots which can perform several tasks and process more comprehensive applications. Another aspect of using GA in robotics is in navigation process. GA provides optimized solutions for navigation process by which the robot can reach to its required destination without being lost or hitting other objects in the environment. In navigation algorithm each chromosome represents a series of path nodes where each node is a gene. Each gene has an x and y value and a Boolean value which represents whether the next node reachable form the current node or no. By using this approach robot will not only be able to always find a way to the target without hitting the objects but also it will be able to find the most optimal path.

2.1.3 Engineering design

Designing a new engineering model is a complex and time consuming process, but designing an optimal model which uses the minimum resources to deliver the maximum output is even much complex. Such a task requires great deal of effort and experience to be completed perfectly. This is where one more time the functionality of Genetic algorithm comes into action. GA can be integrated into computer based engineering design applications. By following such a strategy the application will be able to analyse different aspect of engineering design principles when generating a new design for a given problem. This approach in addition to providing the required design will also assist the designers to identify the frailties and possible failure

points of the design. Such an approach is currently being used in many engineering industries such as aerospace, civil, automotive, robotics, electronics, mechatronics etc. These are a small subset of the fields where Genetic algorithm is currently being used to improve the outcome to the fullest. There are many more fields such as telecommunication routing, trip and shipment routing, gaming, encryption and code breaking, chemical analysis, finance strategies, marketing strategies, which also uses the GA. In summary, Genetic algorithm has obtained a great role in modern world's technological and scientific fields and this is on the increase.

3 CLASSIFICATIONS OF GENETIC ALGORITHMS

A metaheuristic is a procedure which aims to find an acceptable solution in very complex optimization and search problems. [2]

In comparison to other heuristics, metaheuristics make usage of low-level heuristic or search algorithms. Therefore, metaheuristics use concrete heuristics or algorithms which are more abstract.

In comparison to optimization algorithms and iterative methods, retrieved solution is dependent on the set of random variables generated.

Compared to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems. Many metaheuristics implement some form of stochastic optimization, so that the solution found is dependent on the set of random variables generated.

Metaheuristics such as genetic algorithms are characterized by the following attributes [4]:

- Metaheuristics guide search process with defined strategies. Although randomized, metaheuristic algorithms such as GA are not random searches. They exploit prior information to direct the search into a region of better performance within the search space.

- Their goal is to efficiently search the state space to find near-optimal solutions. Thus metaheuristic searches include simple local search algorithms and complex learning processes.

- Metaheuristic algorithms are approximate and usually non-deterministic.

- Metaheuristic algorithms are not problem-specific. They make few assumptions about the optimization or search problem being solved, and therefore they can be used for a variety of problems.

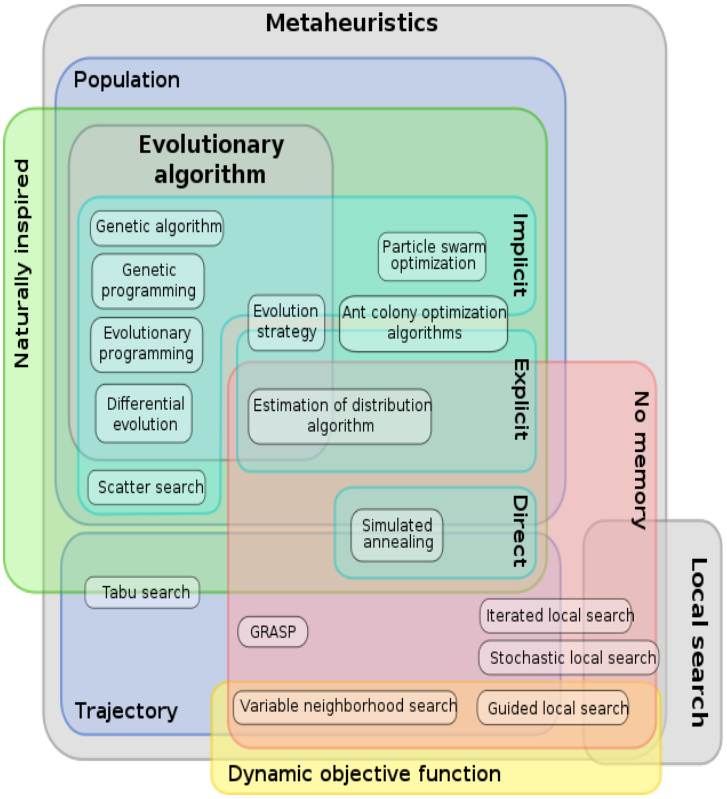

Figure. 1. Metaheuristics classification [5]

GA's are parallel, population-based searches which have a learning component. GA uses stochastic solutions of individual solutions, stochastic crossover and mutation.

3.1 Representation of Genetic Algorithms Using Bit Strings

To represent chromosomes in a genetic algorithm population, strings of bits are normally used, although there are many other existing methods currently in their infancy and under development such as fix-point binary representation, order-based representation, embedded list, variable element list and LISP S-expression.

Every position (locus) in the string (chromosome) has two possible values (alleles): 0 and 1. A solution (chromosome) is composed of several genes (variables). A chromosome represents a solution in the search space of potential solutions.

A population contains a set of chromosomes which are composed of genes. In other words, the solution space contains a set of strings which are composed of bits.

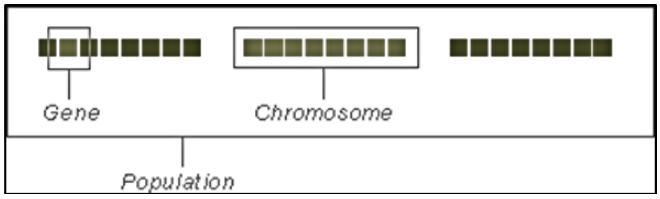

Figure. 2. Population, Chromosomes and Genes [6]

3.2 Implementation Overview

The initial population is first generated at random. Then, the GA goes through the following operators:

3.2.1 Selection operator

The principle here is "survival of the fittest". This operator chooses chromosomes in the population for reproduction. Based on an evaluation function, the 'better' chromosomes are identified and will be more likely to be selected for reproduction. This fitness operator's implementation is dependent on the specific problem at hand.

The 'better' chromosomes may be determined by an objective function which applies uniformly to all chromosomes or by a subjective function, where some rules may not be applied uniformly.

Generally, the probability of selection is proportional to the fitness. It is possible that the best chromosome does not get selected in one run of the GA. However, if the GA is run many times, the probability of selection will converge towards its mathematical expected value.

3.2.2 Crossover operator

This process is closely related to the reproduction of haploid (single-chromosome) organisms or by the fusion of a sperm cell with an ovum to produce a diploid zygote. This operator chooses a position in the string at random. Then, the subsequences before and after that position are exchanged between the strings. The result is the creation of two children strings, i.e. chromosomes.

Given two strings which represent chromosomes, e.g. 11000 and 00111, a crossover operator will randomly select a position, e.g. second position. The resulting children of the crossover will be 11111 and 00000.

Parents: **11|000** and **00|111**
Children: **11111** and **00000**

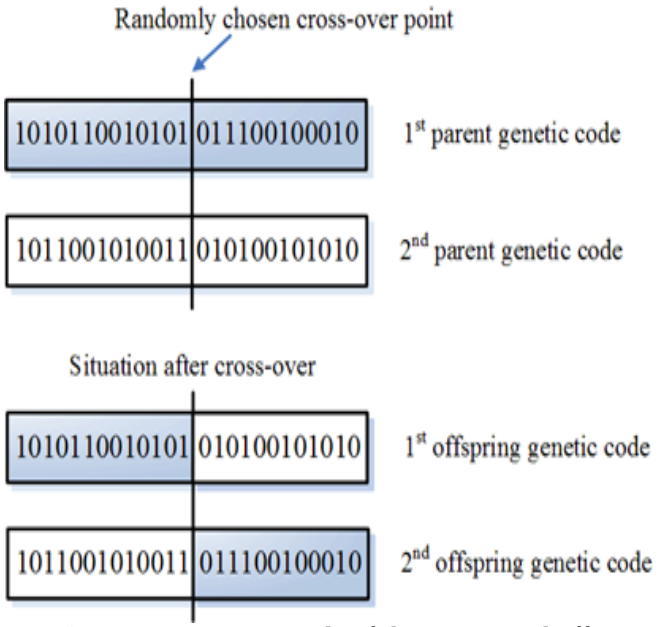

Figure. 3. Genetic Code of the parents and offspring before and after the crossover [7]

Multi-point crossovers are simply crossovers with more than 1 position where crossover will occur.

3.2.3 Mutation operator

This process mimics diversity within a population and helps to prevent premature convergence. This is because mutation and selection alone create hill-climbing algorithms. This operator randomly changes one or some bits in the string representation of a chromosome. For example, a string 10101010 could be mutated in its first position to give 00101010. The probability of mutation and the position of the mutation are controlled by the mutator operator.

Before mutation: 10101010
After mutation: 00101010

3.2.4 Inversion operator

This operator changes the order of genes between two randomly selected points in a chromosome. This fourth operator is rarely used. Its advantages, if any, are not well established. It has been suggested by some artificial intelligence developers that an improvement of search is achieved by such an operator. However, we have noted that all references used for this report indicate that there is no mathematical and empirical evidence of such improvement.

3.3 Effect of Genetic Operators

Using selection alone will have the tendency to generate a population with the fittest chromosomes. Without crossover and mutation, the solution found will be much less optimal.

Using selection and crossover operators only, the GA will converge on a local maxima, which is less optimal than what would otherwise be achieved using all three operators. Using mutation alone makes the GA go on a randomized search through the search space. Using selection and mutation alone will result in the creation of a hill climbing algorithm.

3.4 Pseudo Code for Simple Genetic Algorithm

Assuming that the problem that been clearly defined and candidate solutions have been generated and are represented using X bits, the following steps describe what a genetic algorithm will do, in the following order:

1) Randomly generate n X-bit strings
2) Evaluate fitness of each string in the population using evaluation function f(x)
3) While n offspring have not been created, do
   (a) Select two parent chromosomes based on probability of selection. The higher the fitness, the higher the selection probability.
   (b) Given a crossover and multi-point crossover probability, the crossover operator creates two offspring. In the event no crossover takes place, then the two offspring are identical replicas of their respective parents.
   (c) The two offspring are mutated at each locus using the mutator operator, given a defined mutation probability. The resulting children chromosomes are inserted in the population
   (d) If n is odd, a discard function will have a higher probability of discarding a less fit chromosome
4) The current population is replaced with the new one.
5) Go back to step 2 until fit enough solution is found or until X number of iterations have occurred.

Please note that every time the algorithm goes back to step 2, a new generation is created.

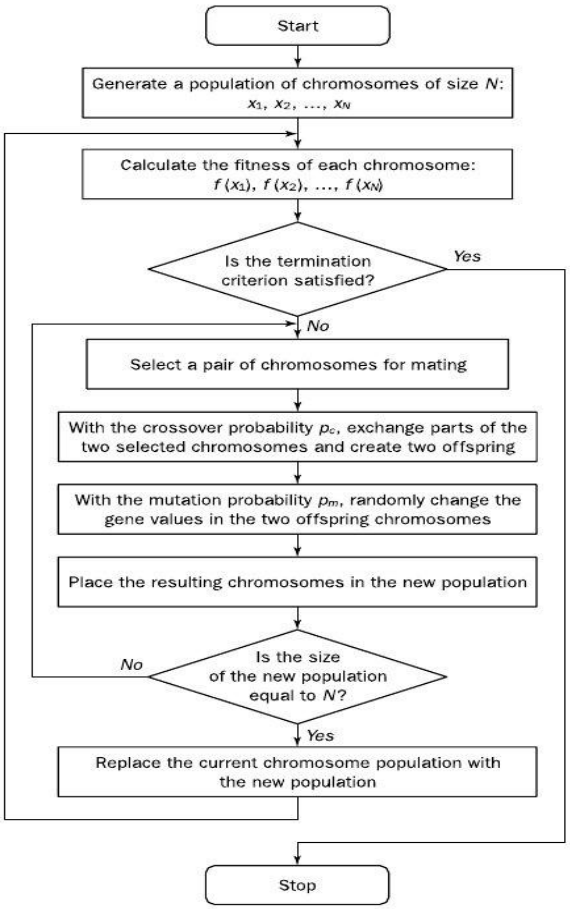

Figure. 4. A basic genetic algorithm [8]

3.5 Genetic Algorithm Parameters

The fundamental parameters of any genetic algorithm are discussed in the following sub-sections.

3.5.1 Cross-over probability

Cross-over points are randomly selected within the length of the chromosomes.

3.5.2 Number of cross-over points

In general, if the chromosome is long, then multiple cross-over points are preferable.

3.5.3 Mutation probability

This refers to the probability of any individual gene which a chromosome to change value from 0 to 1 or vice versa.

3.5.4 The maximum number of iterations

Acknowledging the fact that computational resources are limited, genetic algorithms must have a defined maximum number of generations or a point to indicate when to stop iterating. A maximum number of iterations is especially useful if we have poor knowledge of the ending criteria to indicate the end of iterations.

3.5.5 The value of the termination criterion

If we know how the end criteria should approximately be like, then this parameter is preferred over defining a maximum number of iterations. Examples of termination criteria are:

1. Time Elapsed since fitness increase remains under a certain value

2. If the population diversity falls below a specific value.

3.5.6 The population (generation) size

The greater the number of local maxima, the greater the population size should be. In some cases, the number of local maxima is not known. Therefore the ideal approach would be to have a larger population sample.

4 CONCEPTUAL ADVANTAGE OF GA

Using program evolution to solve computational problems has numerous conceptual applications [10]:

1. To allow a program to adapt and perform according to minimum standards in a changing environment, e.g. robot control in a changing environment or computer interfaces that need to adapt to the idiosyncrasies of an individual user.

2. To allow a program to create innovations on their own, for example, to create a new algorithm to solve a problem. This approach is especially relevant given the fact that intelligent systems are much too complex to program from a ‘top-down’ approach. Therefore an apparently better route to artificial intelligence is through a ‘bottom-up’ approach where programmers encode the simple rules, and complex behaviors (artificial intelligence) are derived from those rules.

3. Other typical search optimization techniques like linear programming, iterations, simple heuristic functions, depth first search and breadth first searches would involve too much time and space complexity for the solution to be economically viable. By searching over a large set of feasible solutions, metaheuristics can often find good solutions with less computational effort. Therefore metaheuristics like GA are useful for optimization problems.

5 CASE STUDY OF GA USING JNETIC

JNetic is software designed by Steve Bergen. It is used to create stylized images from existing ones, by means of a GA [9].

The user can specify the basic GA parameters before running the program. This also includes other settings such as shapes, colors and sizes. However, those other settings will not be varied in this Case study because they are not the focus of this case study is only on GA. The only settings of interest are how changes in GA parameters affect the reproduction of the original image.

JNetic makes use of a base image, and tries to replicate it as best it can using a specified number of shapes and colors, all supplied by the user. We already know how the ‘fittest’ image looks like, all that needs to be done

is to evolve a population to get as close as it can to that image.

The purpose of this case study is to find the parametric values of the GA that provides the most optimum solution (maximum generation, optimum population, optimum crossover rate, optimum mutation rate).

5.1 Selection of Source Image

The image in figure 5 is the SUTS logo, and we will be using this for the experiments.

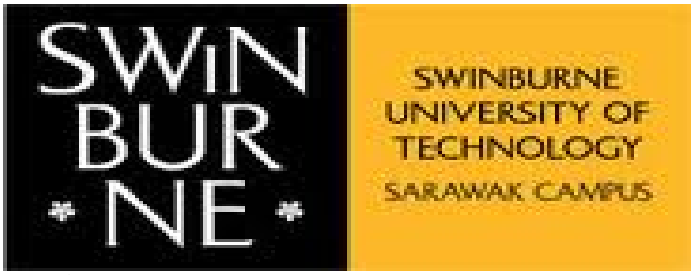

Figure. 5. SUTS Logo

5.2 Choosing Geometric Primitives

The smaller the "pixel" used, the better the final image solution would be. However, the computation resource required to create a solution image which is even grossly similar is extremely high for an average laptop. Therefore, we used large rectangles to represent pixels.

5.3 Choosing a Color Scheme

By default the color model of the geometric primitives will follow a full RGB color scheme, which includes all 16 million colors with alpha transparency, i.e., which have varying degrees of transparency and opacity. A range of transparency allows the creation of graphics that have smoother edges.

Transparency has been disabled due to high resource usage. Therefore, all primitives will have 256 of each red, green and blue value, rendered over a white background.

5.4 Setting GA Parameters

The success of the final image is extremely dependent on the GA parameters used. The recommended number of generations was set to 500, population to 150, crossover rate to 100%, mutation rate to 0.1% and tournament size (number of individuals needed to fill a tournament during selection) to 4.

Image correctness is ranges from 0 to 1, 1 being an exact replica of the original image. Image correctness is measured automatically by the built-in fitness monitor by the program.

Important: Because of the randomness of a GA, every run is likely to produce a slightly different final image

5.5 Changing the Number of Generations

Population = 200,
Crossover rate = 100%,
Mutation = 1%,
Tournament size = 4

TABLE I. NO OF GENERATIONS

Value	Average Image Correctness (5 d.p)	Image
100	0.45238	
200	0.54667	
500	0.71397	
1000	0.79594	
1500	0.81322	

The GA appeared to reach convergence at about 1500 generations, using current parameters.

5.6 Changing Population

Max generation =200, Crossover rate = 100%, Mutation = 1%, Tournament size = 4

A lower population dramatically decreases computational time. The tradeoff however, is a far below optimum solution. The most optimum

population was 100. After this point, there was a 'diminishing returns' on the ratio at an increasing rate.

TABLE II. POPULATION

Value	Average Image correctness (5 d.p)	Computation time/s	Ratio (correctness/time)	Image
50	0.58707	0.077	7.624	
100	0.69151	0.139	4.974	
150	0.71944	0.199	3.615	<negligible difference>
200	0.72498	0.245	2.959	<negligible difference>
250	0.72589	0.304	2.388	<negligible difference>

5.7 Changing Crossover Rate

Max generation = 200, Population = 200, Mutation = 1%, Tournament size = 4

TABLE III. CHANGING CROSSOVER RATE

Value	Average Image Correctness (5 d.p.)	Image
10	0.69028	
30	0.64933	
50	0.62929	
70	0.65025	
90	0.69936	
100	0.64544	

From the above table, the best crossover rate is about 90%. The value of 100% yielded diminishing returns.

5.8 Changing Mutation Rate

Max generation = 200, Population = 200, crossover rate = 100%, tournament size = 4

TABLE IV. CHANGING MUTATION RATE

Value	Average Image Correctness (5 d.p.)	Image
0.1	0.66306	
0.2	0.70309	<negligible difference>
0.4	0.72663	<negligible difference>
0.6	0.73970	<negligible difference>
0.8	0.7399	<negligible difference>
1.0	0.74140	
1.2	0.73669	<negligible difference>
1.5	0.71198	<negligible difference>
2.0	0.67763	

From the above table, it is observed that the best mutation rate is about 1.0 %.

After 1200 generations, population size =100, crossover rate = 90%, mutation rate = 1.0% and using large "pixels" of 10 by 30 to represent rectangles, the best possible result is shown in figure 6.

It is possible to achieve near 100% replica if the pixel sizes used were much smaller. However, the

population sizes and generations must be increased and this demands days of computational power on a laptop.

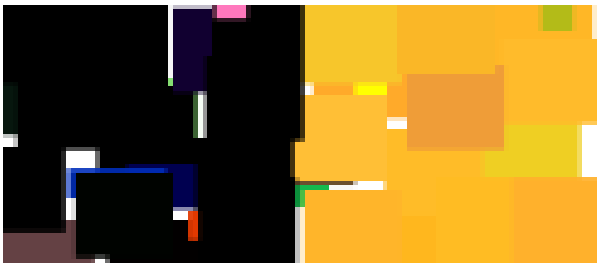

Figure. 6. FINAL SUTS Logo

A progress chart labels the best, worst and average fitness over time. Once the average and best finesses reach near the same value, the GA has converged, and cross-over will be near-useless. This made a compelling case to increase mutation rate to maintain diversity.

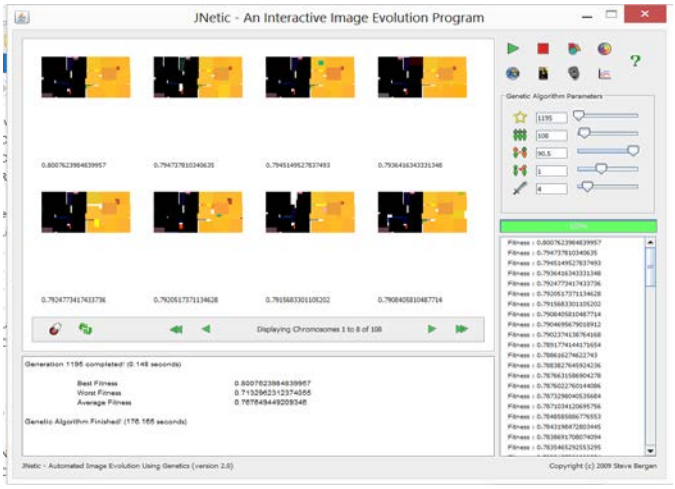

Figure. 7. JNetic Software

6 CONCLUSION

In this paper, we have implemented a case study to find the parametric values of the GA that provides the most optimum solution (maximum generation, optimum population, optimum crossover rate, optimum mutation rate). We have used JNetic software tool that is Textures system use for evolutionary image generation. This application used genetic programming to generate the images with unique feature for texture formulae and vector graphics.

In conclusion, Genetic Algorithm is an exhaustive approach which can be applied in Artificial intelligence field to find optimal solution in complex search spaces. It is a heuristic search algorithm that will exploit the historical information for best solution. The genetic algorithm is well suited for high complexity problems without any known sophisticated solution techniques like the combinatorial optimization problems. By applying the comprehensive functionalities of Genetic Algorithm in real world scientific and industrial fields, it will not only be ensured that many of the existing problems will be resolved but also there will be a promising future towards development of agents which can perform such a task efficiently and effectively without human intervention.

REFERENCES

[1] A. E. Eiben (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: 78–87. ISBN 3-540-58484-6.

[2] M. Melanie (1999), "An Introduction to Genetic Algorithms", Cambridge, MA: MIT Press. ISBN 9780585030944.

[3] C. Blum & A. Roli (2003). "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys 35.3, pp. 268–308.

[4] J. Dréo (2011), "Metaheuristics classification.svg", The Free Encyclopedia. Wikimedia Foundation Inc, http://en.wikipedia.org/wiki/File.

[5] T. Wong & H. Wong, "Introduction to Genetic Algorithms", Department of Computing Imperial College of Science and Technology and Medicine UK, Accessed on 10 September 2013, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html.

[6] Prof. Dr. R. Safaric & Dr. A. Rojko, "Genetic code of the parents and the offspring before and after the cross-over", Maribor December 2006, http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/_04.html.

[7] E. Schultz, J. Mellander & C. Endorf (2008), "Intrusion Detection & Prevention - A basic genetic algorithm", [image online], http://my.opera.com/blu3c4t/blog/show.dml/2636486.

[8] S. Bergen, "JNetic Textures", http://www.cosc.brocku.ca/~bross/JNeticTextures/.

[9] Prof. Dr. Riko S, (December 2006), "Intelligent Control Techniques in Mechatronics - Genetic Algorithm", http://www.ro.feri.unimb.si/predmeti/int_reg/Predavanja/Eng/3.Genetic%20algorithm/index.html

[10] A. Varma & Nathan Erhardt, "Artifical Intelligence from a philosophical and biological perspective", http://biology.kenyon.edu/slonc/bio3/AI/index.html.