A Genetic Approach to Parameterization of Feature Extraction Algorithms in Remote Sensing Images

Edmore Chikohora, Obeten O. Ekabua

Abstract - Genetic Algorithms (GA) are an adaptive heuristic search algorithm found on the evolutionary ideas of natural selection. In this paper, we propose an adaptive heuristic based on the Gabor Filter (GF) to generate useful solutions to optimization of parameter selection strategies for Feature Extraction Algorithms (FEA) in Remote Sensing Images. Experiments were done using computer simulations and a critical analysis on performance of the heuristic algorithm is done in a comparative manner with the rest of the algorithms.

Index terms: Average Ranking, Square Error, Local Extrema, Phenotype, Genotype.

I. INTRODUCTION

Remote Sensing Images (RSI) employs various Feature Extraction Techniques (FET) such as the Gabor Filter (GF), the Modified Gabor Filter (MGF), Mathematical Morphology (MM) among other techniques. These techniques are however known to use fixed parameters such as frequency, orientation, angle, direction and distance whose values are determined using empirical experiments [1].

Although experimentation on parameters has produced some desired results, using a genetic approach to determine parameter values can significantly enhance accuracy on the images obtained by FET in RSI.

This study falls under RSI processing which encompasses feature extraction. In feature extraction, there are many areas of specialization that can be looked into, for example, some algorithms have been designed to identify specific target objects while others focus more generally on say buildings or roads extraction and the extraction techniques for these different specializations can vary substantially.

In this paper, we propose an approach called *The GenApp* (derived from Genetic Approach) which implements GA to determine parameter values for FEA using the adaptive principles of the biological genes.

The distinguished feature of this approach is that it initially generates values for the initial population from a range specified by Information Datagram (ID) strategy and then uses these values with the same weight to find a good solution, while mutation and crossover operators' follows to generate permutations of different weights in order to improve the first computed "good" solution.

The GF will be considered as our control and as a result, a detailed analysis on the functionalities, limitations and cited examples of the GF technique will be discussed as background to our study.

Manuscript Received on May, 2014.

Edmore Chikohora, Department of Computer Science North-West University, Mafikeng Campus Private Bag X2046, Mmabatho 2735, South Africa.

Obeten O. Ekabua, Department of Computer Science North-West University, Mafikeng Campus Private Bag X2046, Mmabatho 2735, South Africa.

The rest of the paper is organised as follows: section II provides a background study of the GF, section III discusses genetic operators, section IV presents parameterization using the proposed GenApp approach, section V provides experiments and their results and section VI concludes the study with a discussion on future work.

II. BACKGROUND STUDY

Parameter selection plays a crucial role in the use of FET as the choice of "good" parameters gives a "good" output image [2] [3]. In this section we review how different authors discussed parameter selection strategies in their publications.

A. The Gabor Filter

Moreno et al. in [3] explored ID, a strategy that selects filter parameters of the GF semi-automatically. However, the strategy only looked at three parameters, that is, orientation (∅), aspect ratio (γ) and sigma of the Gaussian envelope (σ). The rest of the parameters were specified using data obtained from empirical experiments.

The ID strategy worked by initially looking at local extrema for each selected parameter ie. ∅, then compute the highest local maximum and smallest local minimum using formulae (1)-(6);

Highest Local Maximum:

$$(\sigma_i^{max}, \gamma_i^{max}) = arg\ max_{\sigma,\gamma}\ \emptyset - ID_{x,y}^{\emptyset_i} \quad (1)$$

Smallest Local Minimum:

$$(\sigma_i^{min}, \gamma_i^{min}) = arg\ min_{\sigma,\gamma}\ \emptyset - ID_{x,y}^{\emptyset_i} \quad (2)$$

From 1 and 2, the set of chosen GF parameter will be:

$$P_{\emptyset}^{min} = \{(\sigma_i^{min}, \gamma_i^{min}, \emptyset_i), \dots, (\sigma_n^{min}, \gamma_n^{min}, \emptyset_n)\} \quad (3)$$

$$P_{\emptyset}^{max} = \{(\sigma_i^{max}, \gamma_i^{max}, \emptyset_i), \dots, (\sigma_n^{max}, \gamma_n^{max}, \emptyset_n)\} \quad (4)$$

Consequently, the selected feature vector representation becomes:

$$v_{(x,y)} = (v_{(x,y)}^1, \dots, v_{(x,y)}^i, \dots, v_{(x,y)}^m)^T \quad (5)$$

where,

$$v_{(x,y)}^i = \left|G_{\emptyset_i,\lambda_i,\sigma_i}(x,y)\right|, (\lambda_i, \emptyset_i \sigma_i) \in P_{\emptyset}^{max} \cup P_{\emptyset}^{min} \quad (6)$$

Table 1 illustrate the parameter values for ∅, σ and γ obtained by computing for each ID the distance metrics and the feature model type.

Table 1: Computed values for parameters ∅,γ, σ using Information Datagrams [3], [2].

Test	ID type	# local max	# local min	distance	mag	re+im
1	θ	1	1	Mah	68.49	78.33
2	θ	2	0	Mah	85.92	95.83
3	γ	2	0	Mah	58.19	74.16
4	γ	1	1	Mah	54.41	75.83
5	σ	2	0	Mah	58.19	72.50
6	σ	1	1	Mah	50.21	72.50
7	θ	1	1	Euc	31.93	85
8	θ	2	0	Euc	38.87	87.5
9	γ	2	0	Euc	17.86	53.33
10	γ	1	1	Euc	15.55	45
11	σ	2	0	Euc	24.79	74.17
12	σ	1	1	Euc	15.97	75.83

B. The Modified Gabor Filter

Yang et al. [4] used an approach that decomposes period T of the GF into periods T_1 and T_2 which were later specified as in (iii). The remaining parameters $\emptyset$ and σ_y were specified as elaborated in (i) and (ii) respectively.

(i) Specifying Orientation $\emptyset$

The original image is divided into blocks of sizes W x W and the parameter $\emptyset$ is specified as the orientation of each pixel in the block using the following formula;

$$\emptyset(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{u=i-w/2}^{i+w/2}\sum_{v=j-w/2}^{j+w/2} 2G_x(u,v)G_y(u,v)}{\sum_{u=i-w/2}^{i+w/2}\sum_{v=j-w/2}^{j+w/2}\left(G_x^2(u,v)-G_y^2(u,v)\right)}\right) \quad (7)$$

Where W is the size of each subdivided block, G_x and G_y is the local gradient at each pixel in each subdivided block. The resulting orientation value of $\emptyset(i,j)$ obtained from (7) is later regularised into the range $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$.

Specifying Standard Deviations σ_x *and* σ_y

The two parameters σ_x *and* σ_y were specified differently owing to their effects on the output image. In [4] σ_y was empirically set to 4.0 while σ_x is of great concern as its performance is related to the periods T_1 and T_2, hence it influences the degree of contrast on an image.

The values of T_1 & T_2 were computed as in (iii) where period T_2 was further subdivided into a smaller range each time T_1 is small. Table 2 (circled) shows some of the σ_x results obtained from the experiments.

Table 2: Computed σ_x values for different T_1&T_2periods [4] [2].

T_1	T_2	σ_x
4	[4, 12]	1.5
4	[14, 18]	1.6
4	[20, 28]	1.8
6	–	1.8
8	–	2.5
10	–	2.7
12	–	3.0
14	–	3.5
16	–	4.0

(ii) Specifying periods T_1 *and* T_2

By analysing figure 1, we can infer that function $F(x;T_1;T_2)$ is a periodic even-symmetric oscillator with the period $(T_1+T_2)/2$, given that from the figure $\frac{T_1}{2}$ and $\frac{T_2}{2}$ corresponds to the regions above and below the x-axis respectively.

The periods T_1 and T_2 were specified by initially identifying a pixel that needs to be filtered, say, $P_a(x,y)$ and noting its colour intensity. This was followed by continuously checking the neighbouring pixels until you find a pixel with different colour intensity and denote it $P_b(x_1,y_1)$.

T_1 was set to $2W_1$ the distance between $P_a(x,y)$ and $P_b(x_1,y_1)$ while T_2 was set to $2W_2$, that is, the distance between $P_b(x_1,y_1)$ and the next pixel $P_c(x_2,y_2)$ with a colour intensity different from $P_b(x_1,y_1)$. Figure 1 shows how the periods T_1 *and* T_2 were calculated on a fingerprint image obtained from an image database [4].

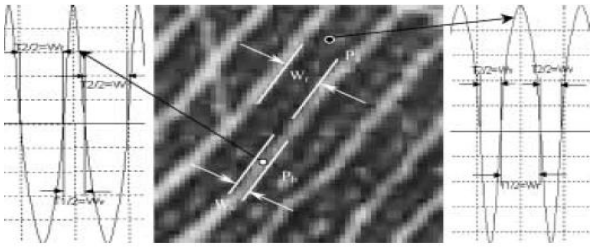

Figure 1: The Curve of $F(X;T_1;T_2)$ Corresponding To Different Periods $T_1$and T_2 [4], [2]

III. THE GENETIC OPERATORS

GA were first proposed by Holland as directed random search techniques which can find the global optimal solution in complex multi-dimensional search spaces [5]. They are known to employ different genetic operators to manipulate individual solutions in a given set of solutions (population) over several iterations (generations) to gradually improve their fitness. This has made them to be successfully applied in many engineering and optimization problems. There are three basic genetic operators that are used to generate and explore the neighbourhood of a population and select a new generation. These are *selection, crossover and mutation* [5]. The operators are applied after an initial population of say, $\boldsymbol{N}$ solutions has been generated and each solution $\boldsymbol{S}$ in the currently generated population is evaluated using a fitness value obtained using equation (8).

$$F(s) = 1.5 * SqE_{max} - SqE(s) \quad (8)$$

Where, $F(s)$ is the fitness value of solution S, SqE_{max} is the maximum square error in the current generation and $SqE(s)$ is the current square error (SqE). The SqE is obtained by computing the following equation;

$$SqE = [(\sum_{n=1}^{N}\sum_{d=1}^{D} X_{nd}^2) - (\sum_{k=1}^{K}\frac{1}{Z_k}\sum_{d=1}^{D} SF_{kd}^2)] \quad (9)$$

Where, SF_{kd}^2 is the square of the sum of the d^{th} solution, K is the number of clusters that are used to group chromosomes, N is the number of patterns represented in vector form and D represent the vector dimension.

At this stage different selection methods such as the *stochastic* or *ranking based selection* are used to select individual solutions with a high fitness value from the selected population.

The *crossover operator* work with pairs of selected individual solutions from the population and is defined in different ways. Given, for example a pair of binary coded solutions **111111** and **000000**, the crossover operator could mean swapping only a single bit between the two solutions commonly known as *one-point crossover* (fig. 2b) or swapping several bits in the binary solutions at once (fig.2c). Figure 2(a) illustrates a general overview of the individual solutions before, during and after the implementation of the crossover operator.

(a)

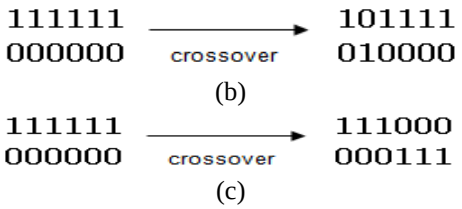

Figure 2: Crossover operator, (a) the crossover overview, (b) the single bit crossover (c) the multiple bits crossover

From figure 2 we can relate that though it's subjective to the selected initial population. The crossover operator can generate large amounts of different possible solutions which may not be quite close to the expected optimum solution or might not cover enough the entire GA's problem space.

As a way to minimize the above gap, a *mutation operator* is performed either on the selection or crossover stage. This will see the elements of the binary solutions being altered from $\mathbf{1}^s$ to $\mathbf{0}^s$ or vice-versa. The mutation operator is implemented using a probability that is kept very low (i.e. 0.001%) [6] as high mutation might end up destroying fit strings (binary solutions) and degenerate the GA into a random walk.

Figure 3 shows a flowchart of a simple GA. The algorithm iterates until a predefined number of generations has been reached.

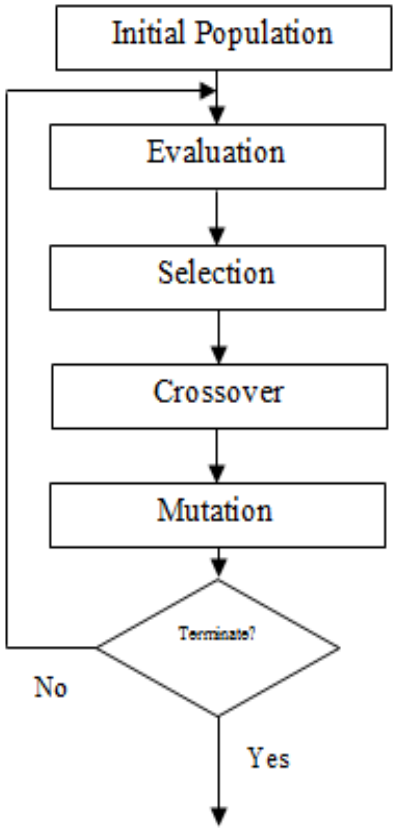

Figure 3: Flowchart: Simplified Genetic Algorithm [5]

IV. PARAMETERIZATION USING GENETIC APPROACH (THE GENAPP)

The main idea of GenApp is to initially select and maintain a population that cover a bigger part of the GA's problem space.

On implementation, the first three stages of the approach (ie. Initial population, Evaluation & Selection) uses *phenotype* which is the physical appearance of the individual possible solution. While the crossover and mutation stages will be implemented using *genotype* being the binary representation of the phenotype. The genetic operators at each stage of figure 3 are implemented as follows;

A. Initial Population

The initial population is generated from the values specified by the ID strategy for the case of $\emptyset$, $\boldsymbol{\sigma}$ and $\boldsymbol{\gamma}$, where we consider the local extrema of each parameter as illustrated below;

(i) For each selected parameter say $\emptyset$*, we first compute local extrema, that is, the highest local maximum* $H(\emptyset)$ *and smallest local minimum* $S(\emptyset)$ *using the ID strategy.*

(ii) Specify the local extrema $H(\emptyset)$ *and* $S(\emptyset)$ *as the upper and lower limits of the initial population for the approach.*

(iii) Randomly generate values from within the extrema range $[S(\emptyset), H(\emptyset)]$ *and consider the generated values as the initial population for the GenApp.*

(iv) Using the fitness formula (8), calculate the fitness value of each member as part of the evaluation process of the initial population.

(v) Using a selection method described in III B. select the highly fit members from the current population that will constitute the next population.

Once the above is achieved, we are guaranteed of an initial population with most of its members having favourable SqE and $F(s)$values.

B. Selection

At the selection stage, the GenApp uses the Average Ranks (AR) ranking method to select possible solutions in the current population that will be passed to the next population. Since the approach uses the minimum SqE technique, we therefore expect members with a small SqE value to have a higher possibility of surviving selection in the current generation [7].

Based on equation 10 the AR generates values that will be assigned to the current population members and for us to properly select the best fit solution, we order the members according to the measured average rank $\boldsymbol{AR_j}$ assigned to each member. The following steps illustrate operation of the AR ranking method;

(i) Let AR_j *be the average rank for each member in a generation, calculate the* AR_j *using the formula 10:*

$$AR_j = \left(\sum_i AR_j^i\right)/n \qquad (10)$$

Where i *is the dataset,* j *is the population member,* AR *is the rank and* n *is the population in a generation.*

(ii) Arrange the average ranks (AR_j) *for the members in the current population in ascending order.*

(iii) Assign ranks starting with the least AR_j *value, such that the best fit member is the one with the least* AR_j *value and assign its rank as 1, the second gets rank 2, and so forth.*

After this stage there is a high probability that the members with the best rankings are closer to the parameter value we are searching and the major function of the remaining genetic operators (crossover and mutation) will be to optimize the final solution.

C. Crossover Operator

The crossover operator mimics propagation and here we implement it by crossing pairs of chromosomes to generate new offspring using the uniform crossover (UX) with a mixing ratio of 0.5. In our approach, the mixing ratio was arrived at based on empirical experiments and analogically the conclusion that in a large search space, a GA that uses UX outperforms a GA that use one-point or two-point crossover [8].

Figure 4 confirms the efficiency of the UX over the one-point and two-point crossover as stated earlier, though the experiments carried out indicate that the UX outperforms all other crossover operators mostly in a larger population size.

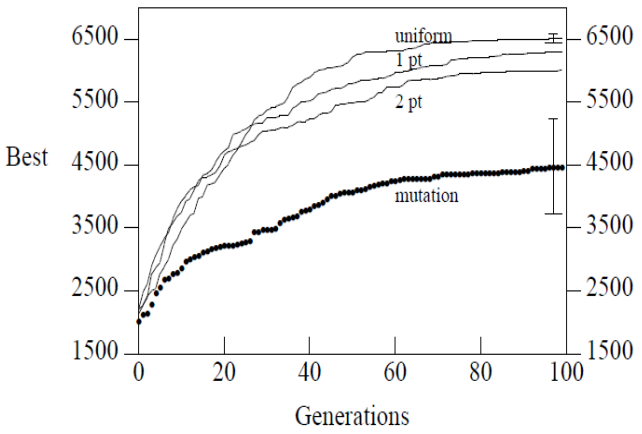

Figure 4: Performance Test of Crossover Operators With a population size 100 Using GenNet [8].

By using a mixing ratio of 0.5, it follows that the generated offspring has approximately 50 percent of its genes from the first parent and the other 50 percentage from the second parent, even though the crossover points are randomly chosen as shown in figure 5.

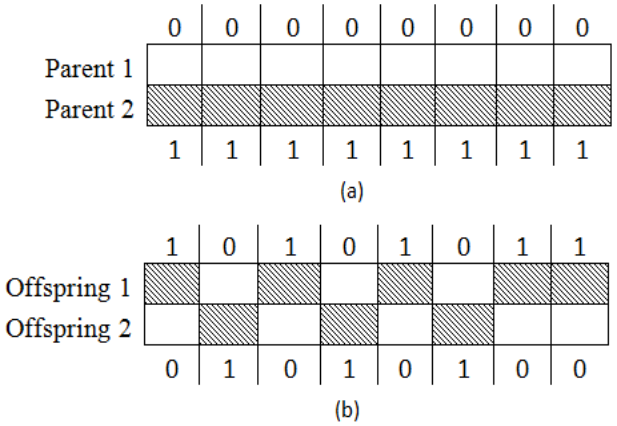

Figure 5: Uniform Crossover with a Probability Ratio Of 0.5, (a) parent chromosomes, (b) offspring after mating with 50% genes from each parent

In our approach we expect the crossover operator to contribute 50 percent of the population for the next generation, while mutation and the Elite chromosomes (parents with best fitness value in current generation) contribute 20 and 30 percent respectively.

D. Mutation

Unlike the crossover, the mutation operator mimics the random changes in nature of the chromosomes within a generation. Since our initial population was derived from a controlled search space, we only consider it for local optimization that is, refining the already generated solution.

As a result, we considered the Random Mutation Hill Climbing (RMHC) to randomly select a neighbor for a candidate solution and mutate only if it improves the current results, otherwise the operator remains unexecuted. Using the RMHC the complete set of features is represented by a binary string of length *N*, where a bit in the string is set to "**1**" if it is to be kept and set to "**0**" if it is to be discarded [9]. The following steps summarises the RMHC method.

*(i) Initialize a binary string S, of length N, where M features are marked as used, '**1**' and the remaining N-M are '**0**'*

(ii) Convert the binary string S to phenotype S^1 and test its fitness $F(S^1)$ using a fitness function (8).

(iii) Randomly mutate M bits in the binary string S.

(iv) Return to step (ii) and continue until either the fitness goal is reached or the maximum number of iterations is reached.

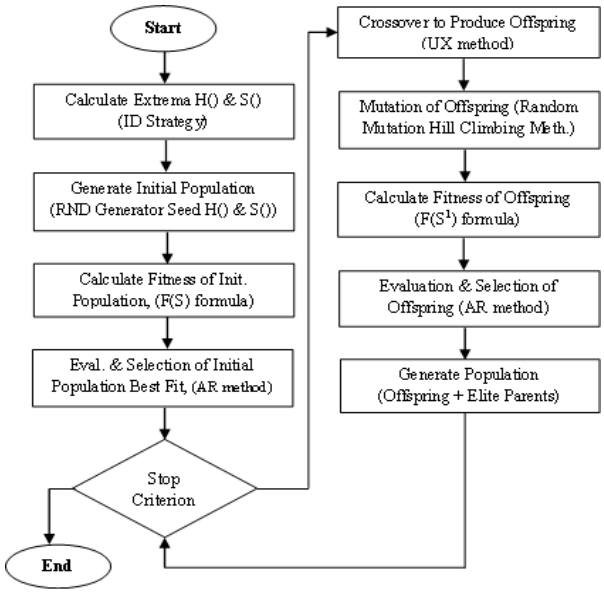

Figure 6: GenApp Flowchart

E. The GenApp Algorithm

1. *Initialize ();*
2. *Compute Extrema (H, S) using ID Strategy (equ. 1-6);*
3. *Generate (P_c), where ($S \leq P_c \leq H$);*
4. *Compute Fitness (F(s) equ.8)*
5. *Evaluate (P_c); // AR ranking method*
6. *Res=BestOf (P_c);*

7. *Repeat_Loop1 until Stop (Criterion)*
8. P_n = 0;
9. *Repeat_Loop2 (P_c)/2 times // UX ratio of 0.5*
10. *Select* $P_1 \in P_c, P_2 \in P_c$;
11. *Crossover* (P_1, P_2, C_1, C_2);
12. $P_n = P_n \cup \{C_1, C_2\}$;
13. *End Repeat_Loop2;*

14. $\forall p \in P_n$, *Compute Fitness (F(s));*
15. P_c = *BestOf* (P_c) ∪ *BestOf* (P_n);
16. *Evaluate* (P_c);
17. $\forall p \in P_c$, *Possible Mutate (p) //RMHC method with a probability of 0.01*
18. *Evaluate*(P_c)
19. *Res = BestOf*($P_c \cup \{Res\}$);
20. *End Repeat_Loop1*
21. *Output (Res);*

In the GenApp algorithm the genetic representation of each individual is formulated by *Initialize ()* and the current population P_c is constructed from randomly generated individuals by *Generate* (P_c). Compute *Fitness (F(s))* uses the SqE technique to calculate the fitness of each individual member P, Evaluate (P_c) assign fitness ranks to the individuals, and *BestOf (P)* finds the individual with the highest fitness value.

Repeat_Loop1 simulates generation cycles while *Criterion* terminates the simulation either through set number of generations or when optimized solution is reached and *Repeat_Loop2* simulates the UX operation. In each generation, a set of offspring P_n of size *P*, will be yielded by the crossover operation *Crossover* (P_1, P_2, C_l. C_2) where P_1 and P_2 are the mating parents yielding two offspring C_1 and C_2 which are added into the current population P_c. *Mutate (P)* uses a *0.01* probability to randomly change the genotype of each individual in P_c while *Output (Res)* gives the final result from the algorithm.

V. EXPERIMENTAL RESULTS

The proposed GenApp approach for parameterization of FET in RSI has been implemented using GenAID, a simulation tool that uses macros in standard Microsoft Excel, coded using Visual Basic language. Ms Excel worksheets were used to capture data for the experiments which include among others, the fitness values and the binary coded chromosomes for the phenotype and genotype respectively, the standard GF parameters as well as producing a fitness chart.

For the purpose of our experiments, three different scenarios were created where genetic operators' parameters were set differently to enable us to draw informed conclusions.

A. Scenario I:

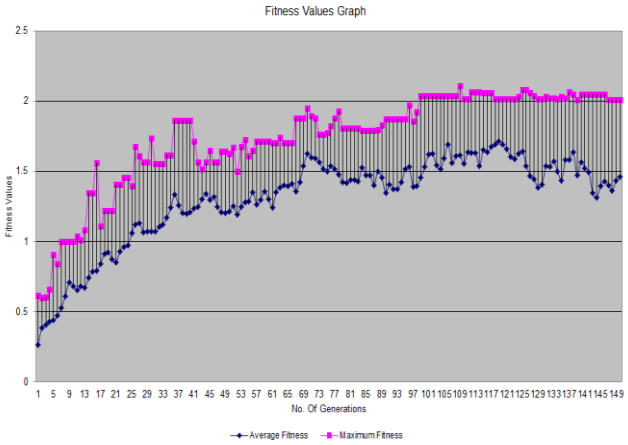

Figure 7: Graphical Representation of Fitness Values Over 150 Generations on Population Size of 284 and Mutation & Crossover Probabilities of 0.01 & 0.5 Respectively.

B. Scenario II:

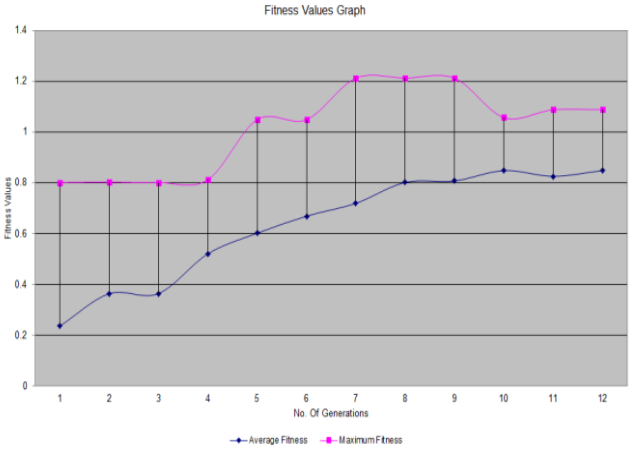

Figure 8: Graphical Representation of Fitness Values Over 12 Generations on Population Size of 284 and Mutation & Crossover Probabilities of 0.01 & 0.5 Respectively.

C. Scenario III:

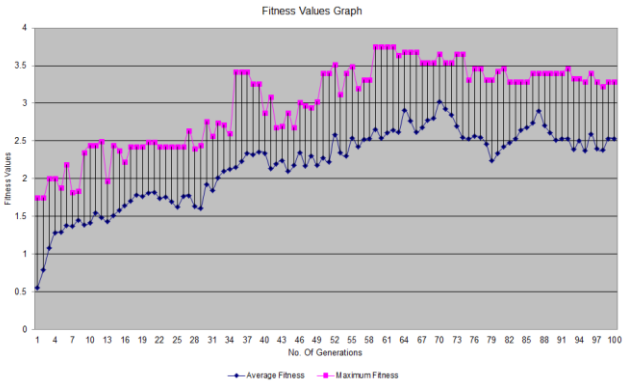

Figure 9: Graphical Representation of Fitness Values Over 100 Generations on Population Size of 20 and Mutation & Crossover Probabilities of 0.01 & 0.5 Respectively.

The figures presented in the above scenarios show two lines representing the average and the maximum fitness obtained over different generations. Average fitness was considered in place of minimum fitness in order to prevent our algorithm from being trapped at a local minimum. Looking at table 3 it's worthwhile to mention that our approach played a greater role in providing an initial population that addresses the problem state space by minimizing the SqE and maximising the fitness value at each generation.

The difference between the initial average fitness (at generation 1) and final fitness (fitness value at the last generation) is reasonably small serve for scenario III where difference is a bit higher resulting from a smaller population size that was used. Otherwise we can note that our algorithm managed to search for a *"good"* solution on the first iteration, leaving the rest of the iterations and genetic operators to do nothing but just optimise the solution.

Table 3: Comparative Analysis of The Results obtained in the 3 Scenarios

Scenarios	Max. Fitness	Average Fitness	Difference
I	2.10	2.03	0.07
II	1.21	1.04	0.17
III	3.75	3.46	0.29

VI. CONCLUSIONS AND FUTURE WORK

The paper looked at genetic operators in a generalized manner, then developing a novel algorithm *"The GenApp"* that implements an adaptive approach in determining parameter values for the FEA in RSI processing. Several experiments were carried out, classified into three different scenarios with a mix of genetic operator values were passed to the simulator in order to obtain a broader view of the novel algorithm. The results obtained were presented graphically and it is noticeable that our approach provided an initial population pool with highly fit members as indicated in table 3. It was also noted that the fitness levels in all the scenarios improved with the growth in generations especially after 100, hence the need to have our population evolve beyond that.

Future work focuses on performance analysis and evaluation of evolutionary computing algorithms in comparison with our novel algorithm in RSI processing.

ACKNOWLEDGEMENT

The authors acknowledges the help rendered by *Mr Aaron P. Masina* for providing the simulator used for the experiments and also *Mrs Teressa T. Chikohora* for the suggestions provided during simulations and preparation of this document.

REFERENCES

1. M. Henrique and G. Easson, Feature Extraction from High-Resolution Remotely Sensed Imagery using Evolutionary Computation. Mississippi, USA: Prof. Eisuke Kita, 2011.
2. E. Chikohora and O. O. Ekabua, "Feature Extraction Techniques in Remote Sensing Images: A survey on Algorithms, Parameterization and Perfomance," International Journal of Soft Computing and Engineering, vol. 4, no. 1, pp. 140-144, March 2014.
3. P. Moreno, A. Benardino, and J. Santos-Victor, "Gabor Parameter Selection for Local Feature Detection," in IBPRIA 2nd Iberian Conference on Pattern Recogniton and Image Image Analysis, Portugal, 2005.
4. J. Yang, L. Liu, T. Jiang, and Y. Fan, "A modified Gabor Filter Design Method for Fingerprint Image Enhancement," Pattern Recognition Letters, no. 24, pp. 1805 - 1817, January 2003.
5. H. N. Al-Duwaish, "Parameterization and Compensation of Friction Forces Using Genetic Algorithms," in Industry Applications Conference, 1999. Thirty-Fourth IAS Annual Meeting. Conference Record of the 1999 IEEE, Phoenix, AZ , 1999, pp. 653-655.
6. C. M. Keet. (2002, May) Homepage of Maria (Marijke) Keet : Genetic Algorithms - An Overview. [Online]. Http://www.meteck.org/gaover.html
7. P. B. Brazdil and C. Soares, "A Comparison of Ranking Methods for Classification Algorithm Selection," in Machine Learning: ECML 2000, R. L. Mántaras, Ed. Porto, Portugal: Springer Berlin Heidelberg, 2000, pp. 63-75.
8. W. M. Spears and V. Anand, "A Study Of Crossover Operators In Genetic Operators," in Methodologies for Intelligent Systems, W. Z. Ras and M. Zemankova, Eds. Charlotte, N. C, USA: Springer Berlin Heidelberg, 1991, pp. 409-418.
9. M. E. Famer, S. Bapna, and A. K. Jain, "Large Scale Feature Selection Using Modified Random Mutation Hill Climbing," in Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on (Volume:2) , 2004, pp. 287-290.