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.

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 (θ), 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. θ, then compute the highest local maximum and smallest local minimum using formulae (1)-(6):

Highest Local Maximum:

\[ \sigma_{\max} = \arg \max_{\sigma, \gamma} \theta - 1 \delta(\theta, \gamma) \]

(1)

Smallest Local Minimum:

\[ \sigma_{\min} = \arg \min_{\sigma, \gamma} \theta - 1 \delta(\theta, \gamma) \]

(2)

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

\[ \sigma_{\max} = \{(\sigma_{\max}, \gamma_{\max}, \theta_{\max}), \ldots, (\sigma_{\min}, \gamma_{\min}, \theta_{\min})\} \]

(3)

Consequently, the selected feature vector representation becomes:

\[ v_{i}(x,y) = (v_{1}(x,y), \ldots, v_{i}(x,y), \ldots, v_{n}(x,y))^{T} \]

(5)

where,

\[ v_{i}(x,y) = \left[ G_{\alpha_i, \beta_i}(\sigma, \gamma) \right]_{i}(\lambda_i, \sigma, \gamma) \in \sigma_{\max} \cup \sigma_{\min} \]

(6)

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

<table>
<thead>
<tr>
<th>Test ID type</th>
<th># Local max</th>
<th># Local min</th>
<th>Distance</th>
<th>Max re-min</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>θ</td>
<td>1</td>
<td>Mah</td>
<td>68.49</td>
</tr>
<tr>
<td>2</td>
<td>θ</td>
<td>2</td>
<td>Mah</td>
<td>85.92</td>
</tr>
<tr>
<td>3</td>
<td>γ</td>
<td>2</td>
<td>Mah</td>
<td>58.19</td>
</tr>
<tr>
<td>4</td>
<td>γ</td>
<td>1</td>
<td>Mah</td>
<td>54.41</td>
</tr>
<tr>
<td>5</td>
<td>σ</td>
<td>2</td>
<td>Mah</td>
<td>58.19</td>
</tr>
<tr>
<td>6</td>
<td>σ</td>
<td>1</td>
<td>Mah</td>
<td>50.21</td>
</tr>
<tr>
<td>7</td>
<td>θ</td>
<td>1</td>
<td>Euc</td>
<td>31.93</td>
</tr>
<tr>
<td>8</td>
<td>θ</td>
<td>2</td>
<td>Euc</td>
<td>38.87</td>
</tr>
<tr>
<td>9</td>
<td>γ</td>
<td>2</td>
<td>Euc</td>
<td>17.86</td>
</tr>
<tr>
<td>10</td>
<td>γ</td>
<td>1</td>
<td>Euc</td>
<td>15.55</td>
</tr>
<tr>
<td>11</td>
<td>σ</td>
<td>1</td>
<td>Euc</td>
<td>24.79</td>
</tr>
<tr>
<td>12</td>
<td>σ</td>
<td>1</td>
<td>Euc</td>
<td>15.97</td>
</tr>
</tbody>
</table>

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.

Published By:

Blue Eyes Intelligence Engineering & Sciences Publication
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 $\varnothing$ and $\sigma_y$ were specified as elaborated in (i) and (ii) respectively.

(i) Specifying Orientation $\varnothing$

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

$$\varnothing(i,j) = \frac{1}{2} \tan^{-1}\left(\frac{x(i,j)w^{2} + y(i,j)w^{2}}{x(i,j)w^{2} - y(i,j)w^{2}}\right)$$

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 $\varnothing(i,j)$ obtained from (7) is later regularised into the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

Specifying Standard Deviations $\sigma_x$ and $\sigma_y$

The two parameters $\sigma_x$ and $\sigma_y$ were specified differently owing to their effects on the output image. In [4] $\sigma_y$ was empirically set to 4.0 while $\sigma_x$ 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 $\sigma_x$ results obtained from the experiments.

(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, $P_0(x, y)$ 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 $P_1(x_1, y_1)$.

$T_1$ was set to $2W_1$ the distance between $P_0(x, y)$ and $P_1(x_1, y_1)$ while $T_2$ was set to $2W_2$, that is, the distance between $P_0(x_1, y_1)$ and the next pixel $P_2(x_2, y_2)$ with a colour intensity different from $P_0(x_1, y_1)$. Figure 1 shows how the periods $T_1$ and $T_2$ were calculated on a fingerprint image obtained from an image database [4].
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 $\emptyset$, $\sigma$ and $\gamma$, 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 $AR_j$ 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:

$$AR_j = (\sum_i AR_i^j)/n$$

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].](image)

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](image)

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^i and test its fitness F(S^i) 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.

E. The GenApp Algorithm

![Figure 6: GenApp Flowchart](image)
1. Initialize (,);
2. Compute Extrema (, ) using ID Strategy (equ. 1-6);
3. Generate (,), where \( S \leq P_i \leq H \);
4. Compute Fitness (equ.8)
5. Evaluate (); // AR ranking method
6. \( Res=BestOf(P_i) \);
7. Repeat_Loop1 until Stop (Criterion)
8. \( P_n = 0; \)
9. Repeat_Loop2 (P,i/2 times // UX ratio of 0.5
10. Select \( P_1 \), \( P_2 \) \( \in \) \( P_c \);
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 (equ.8);
15. \( P_c = BestOf(P_c) \cup BestOf(P_n) \);
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 \( P \) is constructed from randomly generated individuals by Generate (,). Compute Fitness (F(s)) uses the SqE technique to calculate the fitness of each individual member \( P \), Evaluate (P) 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 (\( P_1, P_2, C_1, C_2 \)) where \( P_j \) 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 \( P \), 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

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Max. Fitness</th>
<th>Average Fitness</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2.10</td>
<td>2.03</td>
<td>0.07</td>
</tr>
<tr>
<td>II</td>
<td>1.21</td>
<td>1.04</td>
<td>0.17</td>
</tr>
<tr>
<td>III</td>
<td>3.75</td>
<td>3.46</td>
<td>0.29</td>
</tr>
</tbody>
</table>

### 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