RECOGNITION WITH SEARCH METHODS
CHAPTER 6  RECOGNITION WITH SEARCH METHODS

6.1 Introduction

Telugu character recognition has received much attention due to wide variations in the writing style and similarity in the structure of some of the characters of the script. In the last two chapters the methods for the recognition of handwritten characters using neural networks and support vector machines were presented. For a system to perform well, the training data to train the network must be representative of the input data sets. If the difference between the unknown input and the training data is large, the system may not behave well. Hence there happens to be a need for a diverse training data to the system depending on what all the system might expect in future. As there are innumerable ways in which a character can be written, the process still becomes difficult. Closely observing the capability of human mind in the recognition of handwriting, we find that human beings are able to recognize characters even though they might be seeing that style for the first time. This is possible because of their power to visualize parts of the known styles into the unknown character. So the software for character recognition needs adaptability and the ability to solve problems that cannot be solved using deterministic methods. A stochastic search technique based on evolutionary computing is being proposed. This search method is a powerful technique which mimics natural selection and genetic operators. The power of the search technique comes from the ability to combine good pieces from different solutions and assemble them into a single super solution. This chapter addresses the recognition of handwritten characters using search methodology.

6.2 Search Methods

Search methods or search algorithms broadly speaking is an algorithm for finding an item with specified properties among a collection of items. For problem solving and optimization, the search techniques may be divided into two categories. The first category is deterministic or classical methods and the second category is non-classical or stochastic search.

All search and optimization algorithms that use a single solution update in every iteration and that mainly use a deterministic transition rule are called classical search and optimization methods. These methods use a point by point approach,
where one solution is modified to a different solution in each iteration. The outcome of using a classical optimization method is a single optimized solution which is often arrived at by considering the local information. A unidirectional search is generally performed along the search direction to find the best solution based on a pre specified search direction.

The field of search and optimization has changed over the last few years by a number of non-classical and stochastic search algorithms, particularly those inspired by natural process in physics and biology. Impressive results have been demonstrated on complex practical optimization problems and related search applications taken from a variety of fields, but the theoretical understanding of these algorithms remains weak (Radeliffe et al., 1995). Most commonly used nontraditional search approaches are simulated annealing and genetic algorithms.

### 6.2.1 Simulated Annealing

The simulated annealing method resembles the cooling process of molten metal through annealing. At high temperature, the atoms in the molten metal can move freely with respect to each other but as the temperature reduces, the movement of the atoms gets restricted. Finally the atoms get ordered and form crystals which have minimum energy. The formation of the crystal depends on the cooling rate. If the temperature is reduced at a fast rate, the crystalline state may not be achieved and the system may end up in a poly crystalline state which may exist in a higher energy state than the crystalline state. Therefore, in order to achieve the absolute minimum energy state, the temperature needs to be reduced at a slow rate. This process of slow cooling is known as annealing.

The simulated annealing procedure simulates the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. Simulated annealing algorithm is a point by point method which begins with an initial point and a high temperature T. A second point is created at random in the vicinity of the initial point and the difference in the function value ($\Delta E$) at these two points is calculated. If the second point has a smaller function value then the point is accepted, otherwise the point is accepted with a probability of $\exp(-\Delta E / T)$. This completes one iteration of the simulated annealing procedure. In the next iteration, another point is created at
random in the neighborhood of current point and the process is repeated to accept or reject the point. The algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in function values is found. The operation is represented in the flow chart given in Figure 6.1.

![Flow Chart for Simulated Annealing](image)

**Figure 6.1: Flow Chart for Simulated Annealing.**
The initial temperature and the number of iterations performed at a particular temperature are the two important parameters which govern the successful working of the simulated annealing procedure. If a large initial $T$ is chosen, it takes a number of iterations for convergence, but, if a small initial $T$ is chosen, the search is not adequate to thoroughly investigate the search space before converging to true optimum.

Another disadvantage of the method is that it requires the solution to be unique. Because of this constraint it is not a good method to use for problems which have more than one comparable energy levels.

6.2.2 Genetic Algorithms

Genetic algorithms are stochastic search and optimization algorithms, which mimic the nature’s evolutionary principles to drive its search towards an optimal solution. One of the most striking difference between the classical search and search with genetic algorithms is that the genetic algorithms use a population of solutions in each iteration instead of a single solution. Since populations of solutions are processed in each iteration, the outcome of the algorithm is also a population of solutions. If an optimization problem has a single optimum, all the population members can be expected to converge to that optimum solution. If optimization problem has multiple optimal solutions, the genetic algorithm can be used to capture multiple optimal solutions in its final population.

The operation of Genetic algorithm begins with a population of random strings representing the decision variables. Each string is evaluated using a fitness function to find the fitness value. The population is then operated by three main operators. They are reproduction, crossover and mutation.

Reproduction, also referred to as selection operator, is usually the first operator applied on a population. The essential idea of the reproduction operator is that the above average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner.

In the crossover operation, new strings are created by exchanging information among the strings of the mating pool. In this operation two strings are picked from the mating pool at random and some portion of the strings are exchanged between the strings. The mutation operator alters a string locally in a hope to create a better string.
The population generated by the above operators creates a new population of points. The new population is further evaluated and tested for termination. If the termination criterion is not met, the population is iteratively operated by the above three operators and evaluated. One cycle of these operations and the subsequent evaluation procedure is known as a generation. The operation is represented in the flow chart given in Figure 6.2.
Figure 6.2: Flow Chart for Genetic Algorithm.
6.3 Proposed Approach

Genetic algorithm is one of the fast growing searching techniques in engineering problems. This chapter addresses the application of genetic algorithms for optical character recognition.

Genetic algorithms are a very good means of optimization. They optimize the desired property by generating hybrid solutions from the presently existing solutions. These solutions are added to the solution pool and may be used to generate hybrids. These solutions may be better than the solutions already generated. All this is done by genetic operators which are defined and applied to the problem.

Genetic algorithms, when applied to character recognition with the above idea, give an excellent means of combining various styles of writing a character and generating new styles. The newly generated styles may happen to match the character better than the existing characters. This phenomenon resembles the power of humans to visualize parts of the known style into unknown characters.

The process of character recognition starts by reading an input monochromatic image of any size, and is resized to 16 x 16 pixels. The resized image is binarized by Otsu’s method as explained in chapter 3. Then the binary image is converted into a single row bit pattern of 1 x 256 which is then matched with the generated population at various stages. The fitness function selects a set of fittest values to generate a new population which is then operated upon by genetic operators to yield a new generation on which again the fitness function is applied to generate a new population. The process is carried out till a threshold limit for error between the input and the fittest solution is obtained.

6.4 Fitness Function

Genetic algorithms mimic the Darwinian theory of survival of the fittest and principles of nature to make a search process. The individual solutions that survive for the next generation are decided by the fitness or goodness which is measured by the fitness function. The fitness value indicates whether the solution should be retained or not in order to generate a new population. The individuals with fitness value above the average fitness value are passed on to multiply and mutate into the next generation.
As genetic algorithms are naturally suitable for solving maximization problems, the fitness function should be designed in a way that the better solution will have higher fitness value than the other solutions. Minimization problems can also be solved by genetic algorithm by some suitable transformation.

An ideal fitness function may correlate closely with the algorithms goal but an important aspect is that it should be computed quickly. The speed of execution is very important because typical genetic algorithms must be iterated many number of times in order to produce a better solution. Three fitness functions were used to look at solutions and make qualitative assessments and hence return the fitness value of the solution while using genetic algorithms. They are,

- Hamming distance
- Cross correlation
- Nearest neighbor

### 6.4.1 Hamming Distance

Hamming distance is a number used to denote the difference in two binary strings of equal length or in other words the number of positions at which the corresponding bits are different. Hamming distance can be calculated by taking the exclusive OR of the two strings. It is given by

\[
F(x) = \sum_{i=1}^{n} x_i \text{XOR} y_i
\]

Where

\[
x_i \text{XOR} y_i = \begin{cases} 
1 & \text{if the cells are different} \\
0 & \text{otherwise}
\end{cases}
\]

The smaller the value of \( F(x) \) the smaller the difference between the solutions and greater the similarity, so a smaller value is preferred in the entire population. So, to convert the genetic algorithm to a maximization problem with high fitness value, the function \( F(x) \) in equation 6.1 can be modified as

\[
F'(x) = \frac{1}{1 + F(x)}
\]
6.4.2 Cross Correlation

The degree of similarity between two solutions can be calculated using a similarity function which measures the number of matrix cells occupied by both the strings. It is given by the formula

\[ S(x) = \sum_{i=1}^{n} x_i \cdot \text{AND} \cdot y_i \]  \hspace{1cm} 6.3

Where

\[ x_i \cdot \text{AND} \cdot y_i = \begin{cases} 1 & \text{if } x_i = y_i \\ 0 & \text{otherwise} \end{cases} \]

Smaller the value of S(x) the less is the common area shared by models X and Y, the largest value over the entire population is preferred.

Taking into account the various degree of misalignment the similarity function in equation 6.3 is modified to obtain the cross correlation coefficient. It is given by the formula

\[ CC(X,Y) = \frac{S(X) \cdot S(X)}{(N_x^* N_y)} \]  \hspace{1cm} 6.4

Where \( N_x \) and \( N_y \) are the number of cells occupied by the models X and Y respectively. A smaller value indicates that the common area shared by models X and Y is smaller. Thus the largest value over entire population is preferred.

6.4.3 Nearest Neighbour

The nearest cell distance \( d(Y_{ij},X) \) is used to measure the distance between \( ij^{th} \) cell of model Y and the nearest cell occupied by model X. It is given by the formula
\[
\begin{align*}
&d(Y_{ij}, X) = \begin{cases}
0 & \text{if } Y_{ij} = 0 \\
\min \left[ (m-i)^2 + (n-j)^2 \right] / X_{mn} \neq 0 & \text{if } Y_{ij} \neq 0 \text{ and } X_{mn} \neq 0 
\end{cases} 
\end{align*}
\]

Where max cols and max rows indicate the maximum number of rows and maximum number of columns in binary image X. A larger value of \(d(Y_{ij}, X)\) indicates a larger distance between the cell \(Y_{ij}\) and the nearest cell occupied by image X. For any pair of images X and Y the measurement in difference is given by

\[
ND(Y, X) = \frac{1}{N_Y} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[ d(Y_{ij}, X) \right]^{1/2} + \frac{1}{N_X} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[ d(X_{ij}, Y) \right]^{1/2}
\]

Where \(N_Y\) and \(N_X\) are number of cells occupied by models Y and X respectively. A larger value of \(ND(Y, X)\) indicates that the cell difference between the models Y and X is greater. Thus the smallest value over entire population is preferred. To convert this to a maximization problem, the function in equation 6.6 can be reformulated as

\[
ND'(Y, X) = \frac{1}{1 + ND(Y, X)}
\]

**6.5 Generation of New Population**

A population of the individual solutions is maintained within the search space of genetic algorithm, each representing a possible solution. In the field of genetic algorithm each possible solution is referred by a chromosome and a single bit is referred by a gene.

Genetic algorithms function by iteratively updating a collection of potential solutions called a population. Each member of the population is evaluated for fitness in each cycle. A new population replaces the old population using three main operators, with the fittest members being chosen for reproduction. The three main operators are: Selection or reproduction, crossover and mutation.

The selection operator identifies the chromosomes that will be reproducing. The fitness function evaluates each of the chromosomes, fitter chromosomes are selected to reproduce. The tournament selection was used for reproduction. The crossover
operator performs recombination by exchanging the genetic material between two chromosomes. Mutation operator randomly changes the bits or digits at a particular locus in a chromosome.

6.5.1 Reproduction with Tournament Selection

Reproduction is the first operator applied on a population. The primary objective of the reproduction operator is to make duplicates of good solutions and eliminate bad solution in a population, while keeping the population size constant. Because of this reproduction operator is sometimes known as selection operator. The selection process is achieved by performing the following tasks:

1. Identify the good solutions in a population
2. Make multiple copies of good solutions
3. Eliminate the bad solution from the population so that multiple copies of good solutions can be placed in the population.

There exists number of ways to achieve the above tasks. The commonly used selection schemes are:

- Proportionate selection
- Ranking selection
- Tournament selection
- Genitor (steady state) selection.

In the tournament selection, tournaments are played between two solutions and the better solution is chosen and placed in the mating pool. While selecting the solution to play the tournament, the solution whose fitness value is above the average fitness value of the generation are considered. The best solution in the population will win and the worse solution will be eliminated from the population. The reason for choosing tournament selection is that it has better or equivalent convergent properties when compared to any other operator that exists in the literature (Goldberg and Deb, 1991).
6.5.2 Crossover Operator

After the reproduction phase is over, the population is enriched with better individuals. Reproduction makes clones of good strings, but does not create new ones. Crossover operator is applied to the mating pool with a hope that it would create a better string. The aim of the crossover operator is to search the parameter space for new solutions.

Crossover is a recombination operator which proceeds in three steps. In the first step random pair of two individual strings is selected for mating from the mating pool created by reproduction. In the second step cross site is selected at random along the string length, and in the third step the information is swapped between the chromosomes.

Parent strings, being crossed, are not any two arbitrary random strings. These strings have survived tournaments played with other solutions during reproduction. Thus they are expected to have some good bit combinations in their string representation. Hence, the created offsprings are likely to be good strings.

Every crossover operation may not create a better solution, but if bad solutions are created they get eliminated in the next reproduction operation and hence will have short lives. But on the other hand if the offspring is good, i.e. good solution produced by the crossover operator, it is likely to get more copies in the next reproduction operation and likely to get more chances to achieve crossover with other good solution in subsequent generations. Thus more and more solutions in the population are likely to have similar chromosomes. There are different types of crossover operation in genetic algorithms. Single-point crossover and two-point crossover have been used in this work.

In a single-point crossover, a cross site is selected randomly along the length of the mated strings and the bits next to the cross sites are exchanged as shown in Figure 6.3.

![Figure 6.3: Single-Point Crossover](image)
Since the knowledge of appropriate site is not known, it is selected randomly. This random selection of cross site may produce enhanced children if the selected site is appropriate, otherwise it results in strings of poor quality. But if good strings are not created they will not survive beyond the next generation because reproduction will not select those strings for the next mating pool.

In a two-point crossover, two random sites are chosen and the contents bracketed by these sites are exchanged between two mated parents as shown in Figure 6.4.

![Figure 6.4: Two Point Crossover](image)

In order to preserve some good strings selected during reproduction, not all strings in the population are used in the crossover. If a crossover probability of $P_C$ is used then $100 P_C\%$ strings in the population are used in the cross over operation and $100(1- P_C)\%$ of the population are simply copied to the new population. The extent of string preservation reduces with the increase of cross sites in the crossover operator and is maximum with single point crossover.

### 6.5.3 Mutation Operator

Mutation is a genetic operator used to maintain genetic diversity from one generation of population to the next generation. Preserving the mutation is very important for the search. The bitwise mutation operator changes a 1 to 0 and a 0 to 1 with a mutation probability $P_m$. 
The mutation operator alters the strings locally by complementing a bit, thereby achieving a local search around the population. It helps the search algorithm to escape from local minima traps, since the modification is not related to any previous genetic structure of population.

The mutation operations are not performed deterministically because the sites of mutations are selected randomly. If bad strings are created they will be eliminated by the reproduction operator in subsequent generations and if good strings are created, they will be emphasized.

Mutation rate is the probability of mutation which is used to calculate the number of bits to be mutated. Simple genetic algorithm with a population size of up to 200 uses a mutation rate varying from .001 to 0.5. In this work the probability of mutation is 0.004 which is approximately equal to $1/L$, where $L$ is the length of chromosome which is equal to 256.

### 6.6 Flow Chart for Character Recognition

The methodology presented works according to the following flow chart shown in Figure 6.5. The summary of the parameters used in the methodology are given in Table 6.1.

The working of the algorithm starts with an initial population. Each member of the population and the input image which is supposed to be recognized are subjected to the fitness function. This fitness function calculates the similarity coefficient or fitness function to each individual. All the individuals whose fitness value is above the average fitness value are subjected to tournament selection to generate the mating pool. To generate the next population, crossover with a probability of 0.5 and mutation with a mutation rate of .004 is applied to the population. The process is repeated till the average fitness of the new generation is less than or equal to that of the old generation or till the maximum number of generations has been reached. Then the individual with highest fitness is identified.

---

**Table 6.1: Parameters for Genetic Algorithm**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial population</td>
<td>400</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.5</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.004</td>
</tr>
</tbody>
</table>
Figure 6.5: Flow Chart for Character Recognition with Proposed Approach
6.7 Results and Discussion

In this chapter classification of handwritten characters with genetic algorithms was presented. This model has been implemented using MATLAB(R2009b). The observations from the results are as follows:

1. Three fitness functions have been discussed in section 6.4. The genetic algorithm was tried with all the three fitness functions. With hamming distance and cross correlation the percentage of characters correctly classified is 70.5 and is less than when compared to the nearest neighbor fitness function where the accuracy is 75.2%. But the time required for identification of a character with nearest neighbor is three times that of the first two methods.

2. The algorithm was implemented with single-point cross over and two-point cross over. The classification accuracy with single-point crossover was found to be high. The confusion matrix and performance metrics with third fitness function and with single point cross over are given below in Figure 6.6 and Table 6.2

![Figure 6.6: Confusion Matrix with Proposed Genetic Algorithms](image-url)
Table 6.2: Summary of Performance Metrics with Genetic Algorithms

<table>
<thead>
<tr>
<th>Class</th>
<th>Accuracy %</th>
<th>Sensitivity %</th>
<th>Specificity %</th>
<th>Precision %</th>
<th>NPV %</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.5</td>
<td>88.33</td>
<td>99.63</td>
<td>96.36</td>
<td>98.7</td>
<td>.92</td>
</tr>
<tr>
<td>2</td>
<td>99.17</td>
<td>100</td>
<td>99.07</td>
<td>92.30</td>
<td>100</td>
<td>.96</td>
</tr>
<tr>
<td>3</td>
<td>98.17</td>
<td>91.66</td>
<td>98.88</td>
<td>90.16</td>
<td>99.07</td>
<td>.91</td>
</tr>
<tr>
<td>4</td>
<td>97.83</td>
<td>88.33</td>
<td>98.88</td>
<td>89.83</td>
<td>98.70</td>
<td>.89</td>
</tr>
<tr>
<td>5</td>
<td>98.83</td>
<td>95</td>
<td>99.25</td>
<td>93.44</td>
<td>99.44</td>
<td>.94</td>
</tr>
<tr>
<td>6</td>
<td>91.67</td>
<td>51.66</td>
<td>96.11</td>
<td>59.61</td>
<td>94.70</td>
<td>.55</td>
</tr>
<tr>
<td>7</td>
<td>90</td>
<td>45</td>
<td>95</td>
<td>50</td>
<td>93.95</td>
<td>.47</td>
</tr>
<tr>
<td>8</td>
<td>96</td>
<td>86.66</td>
<td>97.03</td>
<td>76.47</td>
<td>98.49</td>
<td>.81</td>
</tr>
<tr>
<td>9</td>
<td>90.5</td>
<td>63.33</td>
<td>93.51</td>
<td>52.05</td>
<td>95.82</td>
<td>.57</td>
</tr>
<tr>
<td>10</td>
<td>89.67</td>
<td>41.66</td>
<td>95</td>
<td>48.07</td>
<td>93.61</td>
<td>.44</td>
</tr>
</tbody>
</table>

3. The accuracy of classification for five classes with labels 1, 2, 3, 4, 5 is greater than 95% and the value of F-measure is also high for the same set of classes.

4. The sensitivity and specificity for class with label 2 are 100% but the accuracy of classification is 99.17 because three characters from other classes are classified under this group.

5. The accuracy of classification and F-measure for classes with labels 7 and 10 is low. Due to the similarity in structure of characters in these two classes, the characters may have been misclassified among these two groups.
6.8 Conclusions

An evolutionary based method for character recognition which is different from the techniques presented in previous chapters has been discussed. The methodology presented in this chapter is easy to implement compared to those techniques. Though the accuracy of classification is less than that obtained with RBF and SVM, the accuracy of classification for classes with labels 1,2,5,8 is high when compared to all the previous techniques and the value of F-measure for these classes is almost approaching the value of one. The time required for this approach is very less because the binary images after thinning are considered as input to the genetic algorithms which mainly eliminate feature extraction and dimensionality reduction. The accuracy of classification may be increased by increasing the number of bits used to represent the binary image.