Application of Genetic Algorithm on Certain Problems of Pattern Recognition

A MAJOR THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

BY

BHAWNA GARG

UNDER THE GUIDANCE OF

Prof. Asok Bhattacharyya

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING
DELHI COLLEGE OF ENGINEERING
(UNIVERSITY OF DELHI), DELHI

CERTIFICATE

This is to certify that the thesis entitled ***"Application of Genetic Algorithm on Certain Problems of Pattern Recognition"*** being submitted by Bhawna Garg in the partial fulfillment of the requirement for the degree of Master of Engineering in Electronics & Communication in the Department of Electronics & Communication, Delhi College of Engineering. University of Delhi is a record of bonafide work done by her under my supervision and guidance. It is also certified that the dissertation has not been submitted elsewhere for any other degree.

(Prof. Asok Bhattacharyya)

Head of the Department

Deptt. of Electronics & Communication Engg.

Delhi College of Engineering

Delhi - 110042

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without a mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

I am grateful to **Prof. ASOK BHATTACHARYYA** (HOD, ECE deptt.) for providing me an opportunity to undertake this project. I am highly indebted to Mr. Akash Tayal (Lecturer, IGIT), my co-supervisor for taking keen interest in my work and for their constant monitoring and invaluable guidance and support throughout the course of my project. I profusely thank them for having patience to clear my doubts and channelise my efforts. Their cheerful disposition made my work all the more enjoyable.

(BHAWNA GARG)

Roll No. 14/EC/03

University Roll No. 9112

Delhi College Of Engineering,

(University of Delhi), Delhi

ABSTRACT

The present thesis aims to realize problems of Pattern Recognition using Genetic Algorithm to derive the benefits of genetic algorithm in the area of optimization procedure, for structural pattern recognition. Among the various transform frameworks in which pattern recognition has been traditionally formulated, the structural approach is considered to be the most intensively studied approach, when morphology (topology) of the pattern, is going to be the main classifier of the patterns. The pattern analyzed for this study is in the form a character pattern. As the characters can be well represented in the form of their shape numbers, which is based on the morphology of the character, the structural recognition approach is followed to efficiently recognize the characters. The issues carefully taken care by the recognition system are: definition of pattern classes, sensing environment, pattern representation, feature extraction and selection.

In order to achieve the desired objective, the features of the character are extracted. Then, these features are used to distinguish the classes of the characters.

Genetic Algorithm is then applied with the help of MATLAB to recognize the characters correctly.

Further, the work includes recognition of character strings and characters rotated by some degree.

CONTENTS

CHAPTER 1

INTRODUCTION

In practical pattern recognition problems, a classification function learned through an inductive learning algorithm assigns a given input pattern to one of the existing classes of the system. Usually, the representation of each input pattern consists of features since they can distinguish one class of patterns from another in a more concise and meaningful way than offered by the raw representation. In many applications, it is not unusual to find problems involving hundreds features. However, it has been observed that, beyond a certain point, the inclusion of additional features leads to a worse rather than better performance. Moreover, the choice of features to represent the patterns affects several aspects of the pattern recognition problem such as accuracy, required learning time and necessary number of samples. The main goal of feature subset selection is to reduce the number of features used in classification while maintaining acceptable classification accuracy.

Feature subset selection in the context of practical applications such as character recognition presents a multi-criterion optimization function, e.g. number of features and accuracy of classification. Character recognition has been a very challenging research field to achieve machine simulation of human reading and is a subject of intensive study for the last three decades. In the early days of character recognition, it was restricted to machine-printed and fixed format characters. With growing computational power and electronic equipments, in 1980's, character recognition became applicable for handprint and noisy forms along with machine-printed characters. While exploring many different methods, the use of genetic algorithm to recognize characters has been redefined during the past decade. Genetic algorithms offer a particularly attractive approach for this kind of problems since they are generally quite effective for rapid global search of large, non-linear and poorly understood spaces. Moreover, genetic algorithms are very effective in solving large-scale problems.

The present thesis aims to realize problems of Pattern Recognition using Genetic Algorithm to derive the benefits of genetic algorithm in the area of optimization.

1.1 APPROACH TO THESIS

The character to be recognized is available in the form of '.bmp' image. To recognize the text character present in the character image, features based on the shape of the character are extracted.

A Genetic Algorithm tool is developed, which tries to minimize a fitness function. This tool uses the concept of biological evolution for efficiently optimizing the problem.

Then, a fitness function is generated based on the features extracted above, which aids the genetic algorithm to take the decision.

Those similar looking characters that are not distinguished in the above steps are separately considered for improving the accuracy in the result. For this, some other features are taken into account.

Finally, the tool developed aims to make the recognition of character string and rotated characters possible.

1.2 THESIS OUTLINE

Chapter 2 highlights the various aspects of Pattern Analysis along with possible fields of feature extraction in digital images. Chapter 3 introduces the basic concept of Genetic Algorithm along the various steps involved. Chapter 4 explains the steps carried out for implementation of character recognition using MATLAB tool. Chapter 5 presents the MATLAB results for various kinds of problems in Character Recognition. Chapter 6 concludes the thesis and provides the direction for future study.

CHAPTER 2

PATTERN ANALYSIS

2.1 DIGITAL IMAGE

An image may be defined as a two-dimensional function, f (x, y), where ‘x’ and ‘y’ are *spatial* (plane) coordinates, and the mapping function ‘f’ at any pair of coordinates (x, y)

is called the *intensity* or *gray level* of the image at that point. When 'x', 'y', and the function (intensity) values are all finite, discrete quantities, we call the image a *digital image*.

A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *picture elements*, *image elements*, *pels*, and *pixels*. *Pixel* is the term most widely used to denote the elements of a digital image.

There are numerous ways to acquire images, like camera, scanners, sensing strips, etc. The output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: *sampling* and *quantization.*

2.1.1 Basic Concepts in Sampling and Quantization

The basic idea behind sampling and quantization is illustrated in Fig. below. Figure 2.1(a) shows a continuous image, f (x, y), that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

The one-dimensional function shown in Fig. 2.1(b) is a plot of amplitude (gray level) values of the continuous image along the line segment AB in Fig. 2.1(a). The random variations are due to image noise. To sample this function, we take equally spaced samples along line AB, as shown in Fig. 2.1(c). The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities. The right side of Fig. 2.1(c) shows the gray-level scale divided into eight discrete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The digital samples resulting from both sampling and quantization are shown in Fig.

2.1(d). Starting at the top of the image and carrying out this procedure line by line produces a two-dimensional digital image.

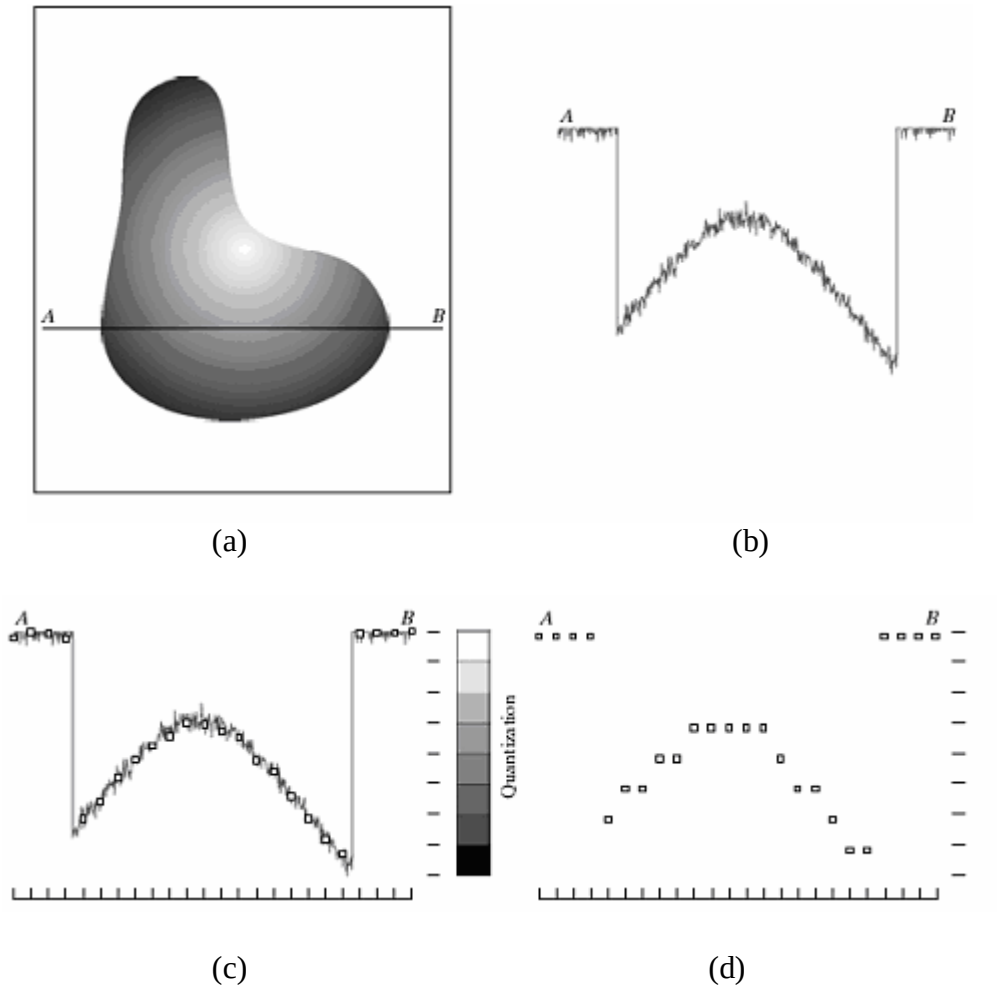


FIGURE 2.1 Generating a digital image. (a) Continuous image. (b) A scan line from *A*to *B*in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

The result of sampling and quantization is a matrix of real numbers. Generally digital images are represented in two principle ways.

1. Assume that an image f(x, y) is sampled so that the resulting digital image has M rows and N columns. The values of the coordinates (x, y) now become *discrete* quantities. For notational clarity and convenience, integer values should be used for these discrete coordinates. Thus, the values of the coordinates at the origin are

(x, y)=(0, 0). The next coordinate values along the first row of the image are represented as (x, y)=(0, 1). It is important to keep in mind that the notation (0, 1) is used to signify the second sample along the first row. It does *not* mean that these are the actual values of physical coordinates when the image was sampled. Figure 2.2 shows one of the possible coordinate conventions that can be used for the representation of digital images.

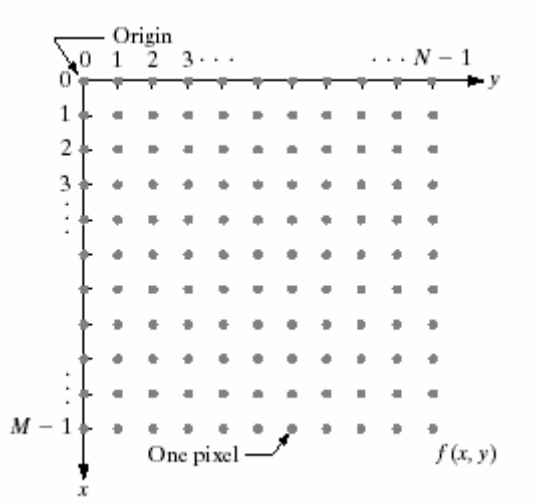


FIGURE 2.2 Coordinate convention used for representing digital images.

Thus, the complete M X N digital image can be written in the following compact matrix form:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}. \qquad \textbf{(2.1-1)}$$

The right side of this equation is by definition a digital image. Each element of this matrix array is called an *image element*, *picture element*, *pixel*, or *pel*. The terms *image* and *pixel* will be used throughout the rest of our discussions to denote a digital image and its elements.

Sometimes, it is advantageous to use a more traditional matrix notation to denote a digital image and its elements:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}. \quad \textbf{(2.1-2)}$$

Clearly, $a_{ij} = f(x = i, y = j) = f(i, j)$, so Eqs. (2.1-1) and (2.1-2) are identical matrices.

2.1.2 Binary Image

Binary Image is the one in which the pixels can have only two values, 1 (for white) and 0 (for black). They are represented by 1 bits/sample. Binary images are not useful for JPEG or MPEG, but they are used extensively in the field of image processing. Especially, binary images help in separating some dominant characteristics of the images. A Binary image can be obtained from a gray level image after performing some pint operation in spatial domain. The term *spatial domain* refers to the aggregate of pixels comprising an image. Spatial Domain methods are procedures that operate directly on these pixels.

2.1.3Morphology

The word *morphology* denotes a branch of biology that deals with the form and structure of animals and plants. The same word is used in image processing in the context of *mathematical morphology* as a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hull.

The language of mathematical morphology is set theory. Sets in mathematical morphology represent objects in an image. For e.g. the set of all black pixels in a binary image is a complete morphological description of the image. In binary images the sets in question are members of the 2-D integer space $\{\mathbb{Z}^2\}$, where each element of a set is a tuple (2-D vector) whose coordinates are the (x,y) coordinates of the black (or white, depending on convention) pixel in the image. Gray-scale digital images can be represented as sets whose components are in $\{\mathbb{Z}^3\}$. In this case, two components of each element of the set refer to the coordinates of a pixel, and the third corresponds to its

discrete gray-level value. Sets in higher dimensional spaces can contain other image attributes, such as color and time varying components.

Few basic operations on binary images, in set theory, can be written as follows:

Difference	$A - B = \{w \mid w \in A, w \notin B\}$ $= A \cap B^c$	Set of points that belong to A but not to B
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \phi\}$	Expands the boundary of A
Erosion	$A \theta B = \{z \mid (B)_z \subseteq A\}$	Contracts the boundary of A

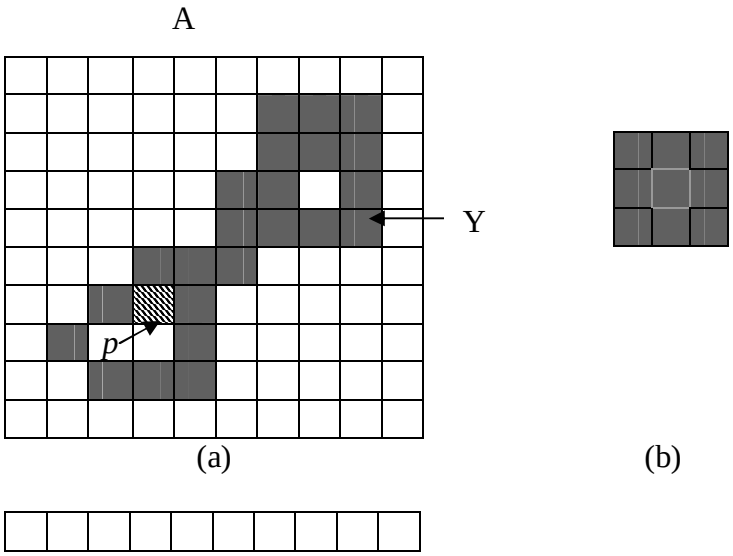
When dealing with binary images, the principal application of morphology is extracting image components that are useful in the representation and description of shape. In particular, we consider morphological algorithms for extracting boundaries connected components, the convex hull, and the skeleton of a region.[2]

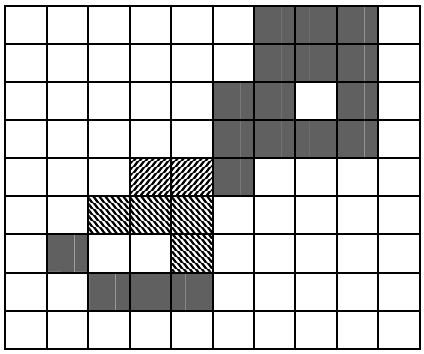
2.1.3.1 Extraction of Connected Components

Extraction of connected components in binary image is central to many automated image analysis and applications. Let Y represent a connected component contained in a set A and assume that a point p of Y is known. Then the following iterative expression yields all the elements of Y:

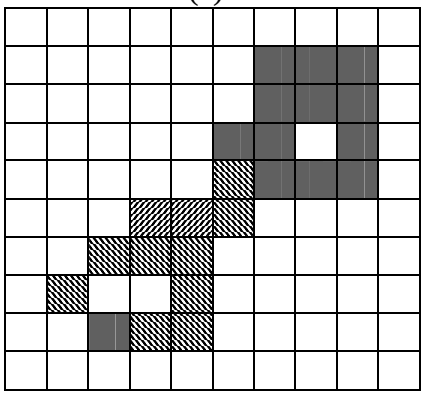
$$X_k = (X_{k-1} \oplus B) \cap A \qquad k = 1,2,3\ldots. \qquad \textbf{(2.1-3)}$$

Where $X_0 = p$, and B is a suitable structuring element, as shown in figure 2.4. If $X_k = X_{k-1}$, the algorithm has converged and we let $Y = X_k$.

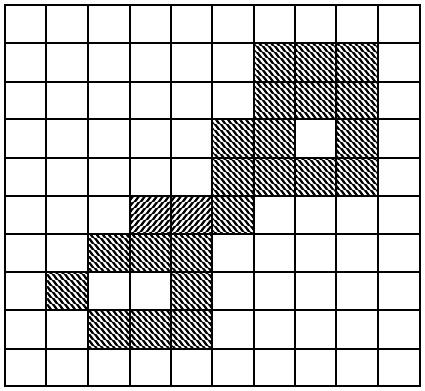




(c)



(d)



(e)

Figure 2.3 (a) Set A showing initial point p (all shaded points are valued 1, but are shown different from p to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

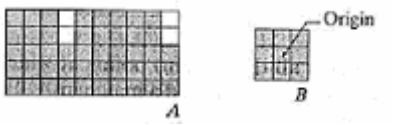
The intersection with A at each iterative step eliminates dilations centered on elements labeled 0. Fig 2.3 illustrates the mechanics of Eq. 2.1-3. Note that the shape of the structuring element assumes 8 – connectivity between pixels.

2.1.3.2 Boundary Extraction

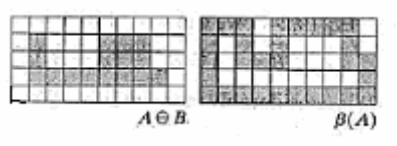
The boundary of set A, denoted by $\beta(A)$, can be obtained by first eroding A by B then performing the set difference between A and its erosion. That is,

$$\beta(A) = A - (A \ominus B) \qquad \textbf{(2.1-4)}$$

where B is a suitable structuring element.



(a) (b)

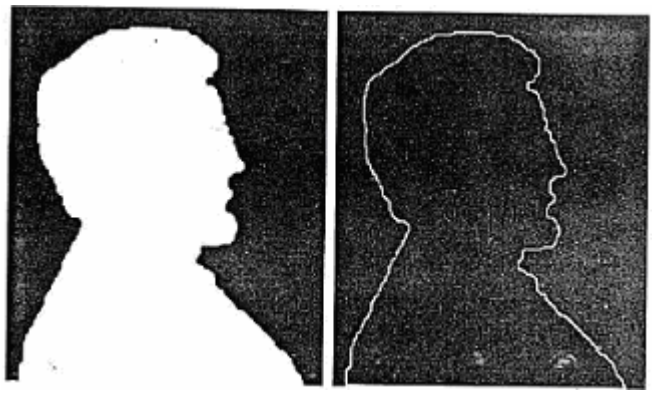


(b) (d)

FIGURE 2.4 (a) Set A. (b) Structuring element B. (c) A eroded by B. (d) Boundary, given by the set difference between A and its erosion.

Fig 2.4 illustrates the mechanics of boundary extraction. It shows a simple binary object, a structuring element B, and the result of using equation 2.1-4. Although the structuring element in fig 2.4(b) is among the most frequently used, it is by no means unique. For e.g. using a 5 x 5 structuring element of 1's would result in a boundary between 2 & 3 pixels thick. The origin of B is on the edges of the set, part of the structuring element maybe outside the image. The normal treatment of this condition is to assume that the values outside the borders of the image are 0.

Fig 2.5 further illustrates the use of Eq. 2.1-4 with the structuring element of Fig 2.4(b). In this example, binary 1's are shown in white and 0's in black, so the elements of the structuring element, which are 1's, also are treated as white. Because of structuring element used the boundary shown in Fig 2.5(b) is one pixel thick.



(a) (b)

FIGURE 2.5 (a) A simple binary image, with 1's represented in white. (b) Result of using Eq. 2.1-4 with the structuring element in Fig. 2.4(b)

From the boundary, the shape of the image can be worked out which is one of the very important feature in pattern recognition.

2.1.4 Shape Analysis

Understanding complex objects using texture, color, motion etc. from the pixel statistics have been widely investigated. Like textures, the form or the shape is another fundamental unit of perception and recognizing objects using their global shape has been gaining immense importance in diverse application areas like biomedical image analysis, video surveillance, biometrics, and so on.

Shape is the ensemble of all the geometrical information of an object which do not change even when the location, scale and orientation of the object are changed. Thus shape is invariant to Euclidean similarity transformations. There are various ways to describe shapes. The technique of shape analysis may be categorized as (1) contour-based shape analysis and (2) region-based shape analysis.

In contour based shape analysis methods, a shape is represented by a coarse discrete sampling of the object contour. A set of landmark points are extracted from this contour.

Since there exists variability amongst the shape patterns, shape preserving transformations, i.e., rigid translation, rotation and scale invariant transforms are necessary before establishing the equivalence amongst the shape classes.

In region based approach, on the other hand, invariant shape features are extracted from the interior as well as boundary pixels inside a region, and classification is carried out using these features.[1]

2.1.4.1 Landmark Points

An interesting way to describe a shape pattern is by defining a finite set of *landmark points* inside the object. Dryden & Mardia define landmarks as points of correspondence on each object that matches between and within the population and classified these points into three subgroups:

1. *Anatomical landmark points:* These points are assigned by an expert that corresponds between organisms in some biologically meaningful way. For example the joints of tissues or bones may be considered as *anatomical*

landmarks. Landmark points have been successfully used for medical image analysis.

2. *Mathematical landmark points:* These are the points located on an object, which obey some interesting mathematical or geometrical property, such as, high curvature or an extremum point.
3. *Pseudo-landmark points:* These are a set of constructed points on an object either on the outline or between other landmark points and may be appropriately defined by the user. Continuous curvilinear shapes may be approximated by a large number of pseudo landmark points. Usually pseudo landmark points may be chosen as a set of equispaced points along the outline between the pair of other landmarks.

Landmark points may be described as nodes or vertices of a polygon enclosing the shape pattern. They may be a set of fiducial points on say biometric patterns like fingerprints. They also may represent some important key points or maker locations in a remotely sensed image, e.g., the intersection between two roads.

2.1.4.2 Polygon as Shape Descriptor

An efficient representation of a planar shape may be concatenation of the ordered pair of an n-input polygon in k dimensions. This can be achieved by concatenating each dimension into a k x n-vector. The vector representation of a two dimensional planar shapes may thus be represented as:

$$x = [\{x_1, x_2, \ldots, x_n\}, \{y_1, y_2, \ldots, y_n\}].$$

The location, scale and rotational invariant shape representation may be achieved by establishing a coordinate reference with respect to position, scale, and rotation, to which all the shape patterns should be aligned.

Quite often we need to align shapes with one-to-one point correspondence. Thus given two shapes, the shape alignment procedure involves four steps:

Step 1. Compute the centroid of each shape.

Step 2. Rescale each shape to have equal size.

Step 3. Align with respect to position the two shapes at their centroids.

Step 4. Align with respect to orientation by rotation.

2.1.4.3 Dominant Points in Shape Description

One scheme to detect a set of dominant points on the curve is to determine the curvature at each point, and computing the resultant cumulative curvature for all the points starting with the last detected dominant point.

Dominant points are those points along an image outline that store a lot of important information about the shape of the image. These points are usually points of high curvature. It does not mean that points of small curvature are devoid of such information. In many cases, the cumulative curvature due to a large number of such points occurring consecutively is significant. A large number of algorithms have been suggested for finding curvature extrema on a digital curve. In general, there are two approaches to the problem.

Method I: To detect the dominant points directly through angle or corner detection schemes.

Method II: To obtain a piecewise linear polygon approximation of the digital curve depending on certain restrictions on the extent to which the shape has been preserved. Dominant points then correspond approximately to the intersections of adjacent line segments of the polygon. These points are also known as the vertices or break points of the closed curve (polygon).

2.1.4.4 Curvature and its Role in Shape Determination

For a smooth curve on a real Euclidean plane, curvature is defined as the change in slope as a function of arc length and can be expressed in terms of first- and second- order derivatives. It is known that points having high curvature are rich in information content regarding the shape of the curve. As a result of this fact, several dominant point detection algorithms use techniques for direct measurement of discrete curvature or its functions (also called measures of significance). Since shape and curvature are intimately related, it is important to find the curvature accurately. Keeping in mind that the curve being used is a digitized version of smooth curve, precise determination of curvature is a challenge. After the determination of digital curvature at each point, the next part is to

detect the dominant points for which several schemes have been suggested. A lot of schemes use the curvature or its functions and then determine the dominant points using a threshold.

2.1.4.5 Polygonal Approximation for Shape Analysis

Given a two-dimensional image the problem of approximating its shape from its polygonal representation has been achieved paramount importance during the last decades. Such a polygonal representation finds a number of applications in diverse areas such as chromosome analysis, industrial machine part classification, character recognition, biometric data analysis etc. The outline of a two-dimensional object usually characterizes the fundamental features of the object patterns. Any closed planar curve may be approximated by a polygon in any desired accuracy so that its representation can have a smooth appearance, which is a sequence of straight-line segments. Identifying such a curve which passes through or near a set of given points is the problem of polygonal approximation.

It has been observed from the human visual information system, that some dominant points along an object contour are rich in information content and they are sufficient to characterize the shape of the object. An approach to apply the basic philosophy of the human understanding and recognition procedure of complex two-dimensional curves leads to detection of dominant points on the curves which when joined by straight line segments can approximate the shape to any desired degree of accuracy.

1. There are many algorithms for the detection of dominant points on digital curves. Piecewise linear approximation of the planar curve allows for a variable number of segments. After an arbitrary initial choice, the segments are split or merged in order to derive the error norms under a prespecified bound.
2. A parallel algorithm is based on the determination of average curvature of the points on the curve by determining the region of support of each point. Since the level of detail represented at each point on the digital curve varies, a smoothing factor based on the local properties of the curve has to be used to find the curvature at each point. This smoothing factor is determined by the region of support. The advantage of the algorithm is that it requires no smoothing parameter.

2.1.4.6 Active Contour Model

Segmentation of monochrome images uses basic properties of gray-level values to detect the isolated points, lines and edges. Alternately segmentation can also be performed by thresholding, region growing, region splitting and merging. Quite often they produce spurious edges and gaps, which do not necessarily correspond to boundary objects. The limitation of these methods is due to their complete reliance on the information contained in the local neighborhood of the image. They ignore both model-based information and higher order organization of the image. Another problem associated with these methods is edge grouping. After extracting edges from the image, they may have to be grouped or linked in order to determine boundaries, which often does not yield good results.

The application of prior knowledge, say geometrical knowledge, strengthens the visual interpretation of shape via the stabilizing influence of prior expectations of the shapes that are likely to be seen. Active contour models are widely used ion detecting object boundary shape as well as they are used for tracking a moving object in an image sequence. A number of approaches have been proposed for shape analysis using active contours. These models utilize deformable contours, which conform to various object shapes and motion. They can be used for edge and curve detection, segmentation, shape modeling and visual tracking.

2.1.4.7 Shape Distortion and Normalization

While capturing two-dimensional images, based on the camera placements, there are 4 possible basic forms of planar object shape distortions-rotation, scaling, translation and skewing. A good shape descriptor should be invariant to these distortions. It is thus important to normalize the shape patterns in its original and various distorted forms, such as scaled, rotated, translated or skewed forms, so that they all, more or less, resemble similar to each other. When an appropriate set of features are extracted only after such normalization, shape classification yields much better accuracy. We will consider shape normalization of a binary image f (x, y), in which f (x, y) = 1 indicates that (x, y) is an object pixel, otherwise it is background pixel. Such a normalization algorithm to normalize the shapes, called shape compacting, involves the following steps:

1. Computing the shape dispersion matrix M,
2. Aligning the coordinate axis with the eigen vectors of M, and
3. Rescaling the axis using the eigen values of M

2.1.4.8 Shape Dispersion Matrix

For a given shape its dispersion matrix M reveals the variances and co-variances amongst the pixels in the image. The dispersion matrix is a key element in the normalization process. The alignment of the coordinate axis uses the dispersion matrix M and takes care of the rotation of the object. Rescaling the coordinate axis is the integral component of shape compacting and it uses the dispersion matrix. The basic philosophy of shape normalization process is that after the normalization operation, the shape will have a dispersion matrix equal to an identity matrix multiplied by a constant. This is an indication that the shape in its most compact form.

To compute dispersion matrix first we calculate the shape centroid

$$\bar{x} = \frac{\sum_x \sum_y x \cdot f(x,y)}{\sum_x \sum_y f(x.y)}, \qquad \bar{y} = \frac{\sum_x \sum_y y \cdot f(x,y)}{\sum_x \sum_y f(x.y)}$$

The shape dispersion matrix M is a 2 by 2 matrix

$$\mathrm{M} = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix}$$

Where,

$$\mathrm{m}_{1,1} = \left(\frac{\sum_x \sum_y x^2 \cdot f(x,y)}{\sum_x \sum_y f(x,y)} \right) - \bar{x}^2, \qquad \mathrm{m}_{2,2} = \left(\frac{\sum_x \sum_y y^2 \cdot f(x,y)}{\sum_x \sum_y f(x,y)} \right) - \bar{y}^2,$$

$$\mathrm{m}_{1,2} = \mathrm{m}_{2,1} = \left(\frac{\sum_x \sum_y x \cdot y \cdot f(x,y)}{\sum_x \sum_y f(x,y)} \right) - \bar{x} \cdot \bar{y},$$

If we consider each object pixel as a data point, the shape can be viewed as a cluster of pixels. The shape dispersion matrix M computed above is exactly the covariance matrix of the cluster. It has already been discussed that a set of principle components may be selected from the covariance matrix, which is used to decouple the set of correlated features. It is also necessary to scale the features so that the clusters become compact. The shape dispersion matrix essentially performs the same function; it normalizes a shape by making it compact.

2.1.4.9 Shifting and Rotating the Coordinate Axes

The origin of the coordinate system is shifted to the center of the shape and then the coordinate system is rotated according to the eigen vectors of the dispersion matrix M.

The matrix M has two eigen vectors E_1 & E_2 corresponding to the eigen values λ_1 & λ_2. The two normalized eigen vectors E_1 & E_2 of M are computed as follows:

$$\lambda_{1,}, \lambda_2 = \frac{\mathrm{m}_{1,1} + \mathrm{m}_{1,2} \pm \sqrt{(m_{1,1} + m_{1,2})^2 + 4m_{1,2}^2}}{2}$$

&

$$\mathrm{E}_1 = \begin{bmatrix} e_{1x} \\ e_{1y} \end{bmatrix} = \begin{bmatrix} \frac{m_{1,2}}{\sqrt{(\lambda_1 - m_{1,1})^2 + m_{1,2}^2}} \\ \frac{\lambda_1 - m_{1,1}}{\sqrt{(\lambda_1 - m_{1,1})^2 + m_{1,2}^2}} \end{bmatrix}$$

$$\mathrm{E}_2 = \begin{bmatrix} e_{2x} \\ e_{2y} \end{bmatrix} = \begin{bmatrix} \frac{m_{1,2}}{\sqrt{(\lambda_2 - m_{1,1})^2 + m_{1,2}^2}} \\ \frac{\lambda_1 - m_{1,1}}{\sqrt{(\lambda_2 - m_{1,1})^2 + m_{1,2}^2}} \end{bmatrix}$$

Now we can construct a matrix R from E_1 & E_2 by:

$$\mathrm{R} = \begin{bmatrix} E_1^T \\ \overline{E_2^T} \end{bmatrix} = \begin{bmatrix} e_{1,x} & e_{1,y} \\ e_{2,x} & e_{2,y} \end{bmatrix}$$

Since M is real and symmetric, E_1 & E_2 are orthogonal to each other. Furthermore, they are normalized to unit length. Thus, R is an orthonormal matrix.

We now transform the coordinate system by first translating the origin to the shape center and then multiplying the coordinates with Matrix R. Now, each object pixel location (x,y) will have a new location (x', y') given by:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \cdot \begin{bmatrix} x - \bar{x} \\ y - \bar{y} \end{bmatrix}$$

Since R is an orthonormal matrix, the geometric interpretation of the transform by R is pure coordinate rotation. The new coordinate axes are in the same directions as E_1 & E_2 .

The dispersion matrix D of the translated and rotated shape is given by:

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

2.1.4.10 Changing The Scales Of The Bases

In the previous steps, we have rotated the coordinate system so that the new X – axis points in the direction in which shape is most dispersed. The effect of the rotation on the dispersion matrix is that now it is a diagonal matrix. Since our objective is to have a shape whose dispersion matrix is a scaled identity matrix, in this last step we will change the scales of the two axes according to the Eigen values λ_1 & λ_2. i.e. for an object pixel location (x', y'), the new location (x'', y'') is obtained through a transformation defined by W:

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = W \cdot \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} k/\sqrt{\lambda_1} & 0 \\ 0 & k/\sqrt{\lambda_2} \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$$

where, k is a system wide constant.

Since W is a diagonal matrix the effect of the above step on the shape is to change the scales of the two coordinate bases vectors so that the shape is in its most compact form and with a normalized size. The results of the translation, rotation and scale invariance compact form generation has been shown in Fig. 2.6.

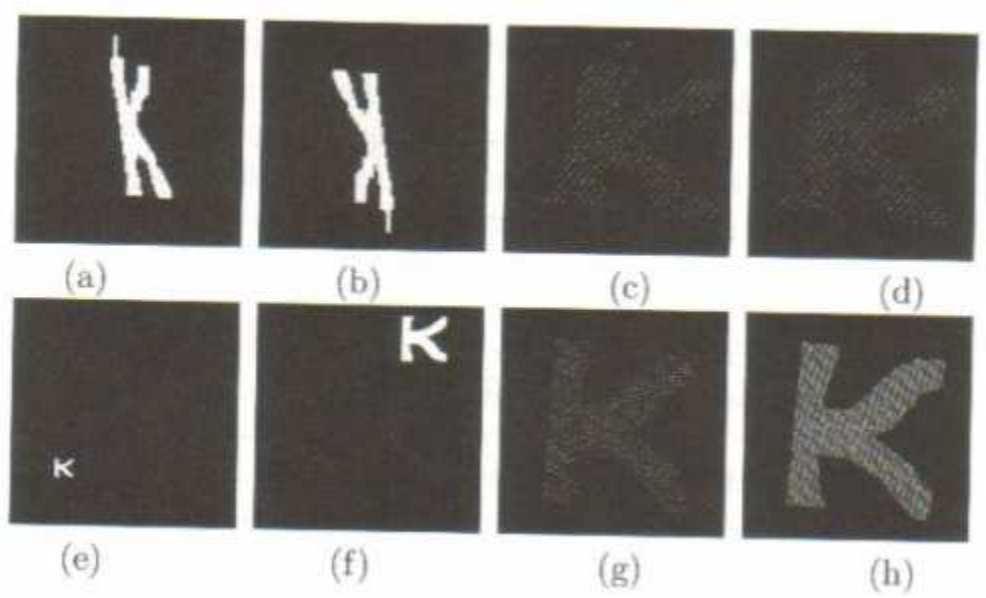


FIGURE 2.6 Normalized images by shape dispersion matrix : (a)-(b) rotated image of letter *K*, (c)-(d) normalized results, (e)-(f) scaled and translated version of letter *K* image, (g)-(h) normalized results.

2.2 PATTERN RECOGNITION

In digital image processing, the process of recognition of *individual* image regions is called pattern recognition. The approaches developed for pattern recognition are divided into two principal areas: decision-theoretic and structural. The first category deals with patterns described using quantitative descriptors, such as length, area and texture. The second strategy deals with patterns best described by qualitative descriptors, such as relational descriptors.

Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.[7]

The act of recognition can be divided into two broad categories: recognizing concrete items and recognizing abstract items. The recognition of concrete items involves the recognition of spatial and temporal items. Examples of spatial items are fingerprints, weather maps, pictures and physical objects. Examples of temporal items are waveforms

and signatures. Recognition of abstract items involves the recognition of a solution to a problem, an old conversation or argument, etc. In other words, recognizing items that do not exist physically.

2.2.1 Pattern

A *pattern* is an *arrangement of descriptors*, such as shape number, signature, number of lines etc. The name *feature* is used often in pattern recognition literature to develop a descriptor. A *pattern class* is a family of patterns that share some common properties. Pattern classes are denoted ω_1 , ω_2 …..ω_W , where W is the number of classes. Pattern recognition by machine involves techniques for assigning patterns to their respective classes –automatically and with as little human intervention as possible.

A **pattern** is an entity, vaguely defined, that could be given a name, e.g.,

- fingerprint image
- handwritten word
- human face
- speech signal
- DNA sequence

The application of pattern recognition studied in this project is *character recognition*. So, character recognition is studied here in detail.

2.3 CHARACTER RECOGNITION TECHNIQUES

Character recognition has been a very challenging research field to achieve machine simulation of human reading and is a subject of intensive study for the last three decades. In the early days of character recognition, it was restricted to machine-printed and fixed format characters. With growing computational power and electronic equipments, in 1980's, character recognition became applicable for handprint and noisy forms along with machine-printed characters.

2.3.1 Steps for Character Recognition

The character recognition process can be grouped into hierarchical task of preprocessing, segmentation, representation, training and recognition. In some methods, some of the stages are merged or omitted; in others a feedback mechanism is used to update the output of each stage.

2.3.1.1 Sensing: Sensing is the process of acquiring an input of character pattern captured from a device like a camera / scanner or extracted from any database that needs to be recognized for analysis.

2.3.1.2 Preprocessing or filtering: The raw data, depending on the data acquisition type, is subjected to a number of preliminary processing steps to make it usable in the descriptive stages of character analysis. Preprocessing aims to produce data that are easy for the CR systems to operate accurately. The main objectives of preprocessing are

1) Noise reduction;
2) Normalization of the data;
3) Compression in the amount of information to be retained;
4) Segmenting the document into its subcomponents.

2.3.1.3 Feature Extraction: Image representation plays one of the most important roles in a recognition system. In the simplest case, gray-level or binary images are fed to a recognizer. However, in most of the recognition systems, in order to avoid extra complexity and to increase the accuracy of the algorithms, a more compact and characteristic representation is required. For this purpose, a set of features is extracted for each class that helps distinguish it from other classes while remaining invariant to characteristic differences within the class. The various types of image representation can be categorized into three major groups:[19]

1) Global Transformation and Series Expansion
2) Statistical representation and
3) Geometrical and topological representation.

2.3.1.4 Model Selection and Training: CR systems extensively use the methodologies of pattern recognition, which assigns an unknown sample into a predefined class. Numerous techniques for CR can be investigated in four general approaches of pattern recognition, as suggested in:

1) Template matching;

2) Statistical techniques;

3) Neural networks (NNs);

4) Structural techniques.

The above approaches are neither necessarily independent nor disjointed from each other. These techniques are discussed in detail in the next section.

The steps of character recognition are summarized in the Fig. 2.7

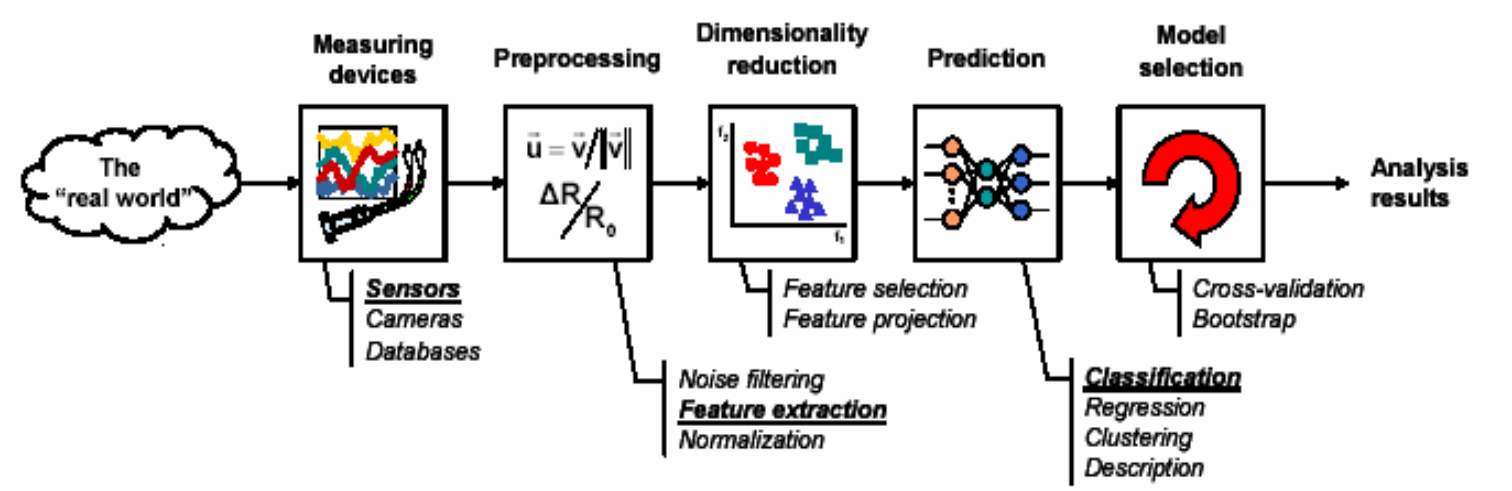


FIGURE 2.7 Modules of a basic character recognition system

2.3.2 Recognition Techniques

Some of the techniques for character recognition are:[8]

2.3.2.1 Template Matching

The conventional method is to consider a region, which includes the letter. If a line passes through a pixel, we give the corresponding pixel a value of 1; otherwise it is taken as 0. Thus we have to store the pixel value combinations for every letter. Now when an input pattern is given, a suitable match with these stored patterns is checked. Only if the new input pattern exactly matches with any of the stored patterns, it will be recognized.

This method has several limitations. First we need to store every pattern that we want to

get recognized in future. So, large memory space is required. Second, it takes a lot of time for matching the input pattern with the stored pattern and producing the results. Third this method cannot handle noise or variation in the input. That is, the input pattern should he exactly similar to one of the stored patterns; otherwise the systems will not be able to recognize it. So this method is poor at generalization.

2.3.2.2 Artificial Neural Networks (ANNS)

A *Neural Network* is defined as a computing architecture that consists of a massively parallel interconnection of adaptive "neural" processors. Because of its parallel nature, it can perform computations at a higher rate compared to the classical techniques. Because of its adaptive nature, it can adapt to changes in the data and learn the characteristics of input signal. A NN contains many nodes. The output from one node is fed to another one in the network and the final decision depends on the complex interaction of all nodes. Because of it's all these characteristics it has become very efficient tool for character recognition.

The drawback of this method is that the size of input vector is very large for large images. There will be a large number of input units and hidden units. So the complexity of the neural network will be large.

2.3.2.3 Statistical Techniques

Statistical decision theory is concerned with statistical decision functions and a set of optimality criteria, which maximizes the probability of the observed pattern given the model of a certain class. Statistical techniques are mostly based on three major assumptions.

a) Distribution of the feature set is Gaussian or in the worst-case uniform.

b) There are sufficient statistics available for each class.

c) Given ensemble of images, one is able to extract a set of features, which represents each distinct class of patterns.

The measurements taken from n features of each word unit can be thought to represent an n-dimensional vector space and the vector, whose coordinates correspond to the measurements taken, represents the original word.[10]

2.3.2.4 Structural Techniques

The techniques discussed above deal with patterns quantitatively and largely ignore any structural relationships inherent in a pattern's shape. The structural methods, however, seek to achieve pattern recognition by capitalizing precisely on these types of relationships.

Matching Shape Numbers

The *degree of similarity*, 'k', between two region boundaries (shapes) is defined as the largest order for which their shape numbers still coincide. For example, let 'a' and 'b' denote shape numbers of closed boundaries represented by 4-directional chain codes. These two shapes have a degree of similarity 'k' if

$$s_j(a) = s_j(b) \quad \text{for } j = 4, 6, 8, \ldots, k$$

$$s_j(a) = s_j(b) \quad \text{for } j = k + 2, k + 4, \ldots.$$

Where s indicates shape number and the subscript indicates order. The *distance* between two shapes 'a' and 'b' is defined as the inverse of their degree of similarity:

$$D(a, b) = 1/k$$

The distance satisfies the following properties:

$$D(a, b) \geq 0$$

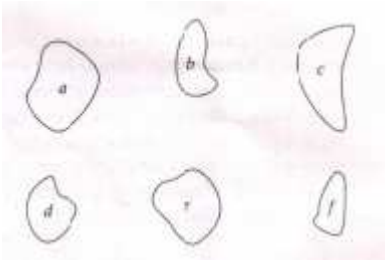
$$D(a, b) = 0 \quad \text{iff } a = b$$

$$D(a, c) \leq \max [D(a, b), D(a, c)].$$

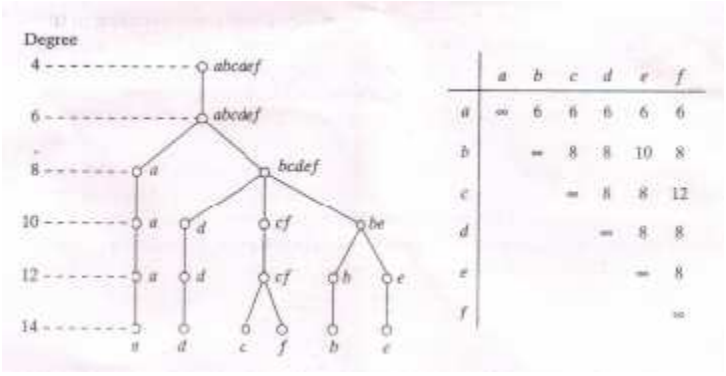
Either 'k' or 'D' may be used to compare two shapes. If the degree of similarity is used, the larger 'k' is, the more similar the shapes are (note that k is infinite for identical shapes). The reverse is true when the distance measure is used.

Suppose that we have a shape *f* and want to find its closest match in a set of five other shapes (a, b, c, d and e), as shown in Fig. 2.8(a). This problem is analogous to having five prototype shapes and trying to find the best match to a given unknown shape. The search may be visualized with the aid of similarity tree shown in Fig. 2.8(b). The root of the tree corresponds to the lowest possible degree of similarity, which, for this example, is 4.

Suppose that the shapes are identical up to degree 8, with the exception of shape ‘a’, whose degree of similarity with respect to all other shapes is 6. Proceeding down the tree, we find that shape ‘d’ has degree of similarity 8 with respect to all others, and so on. Shapes ‘f’ and ‘c’ match uniquely, having a higher degree of similarity than any other two shapes. At the other extreme, if ‘a’ had been an unknown shape, all we could have said using this method is that a was similar to the other five shapes with a degree of similarity 6. The same information can be summarized in the form of a *similarity matrix*, as shown in fig 2.8(c).



(a)



	a	b	c	d	e	f
a	∞	6	6	6	6	6
b		∞	8	8	10	8
c			∞	8	8	12
d				∞	8	8
e					∞	8
f						∞

(b) (c)

FIGURE 2.8 (a) Shapes. (b) Hypothetical similarity tree. (c) Similarity matrix.

Structural techniques can be also applied to handwritten character recognition. In this case a structure can be broken into parts, it can be described by the features of these parts and can also by the relationships between these parts. This can be a either topological or geometrical feature. By choosing proper features each character can be described such that corresponding character is identified clearly.[11]

CHAPTER 3

GENETIC ALGORITHM

3.1 INTRODUCTION

A **genetic algorithm** (**GA**) is a search technique used in computer science to find approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover).

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called *chromosomes*) of candidate solutions (called *individuals*) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

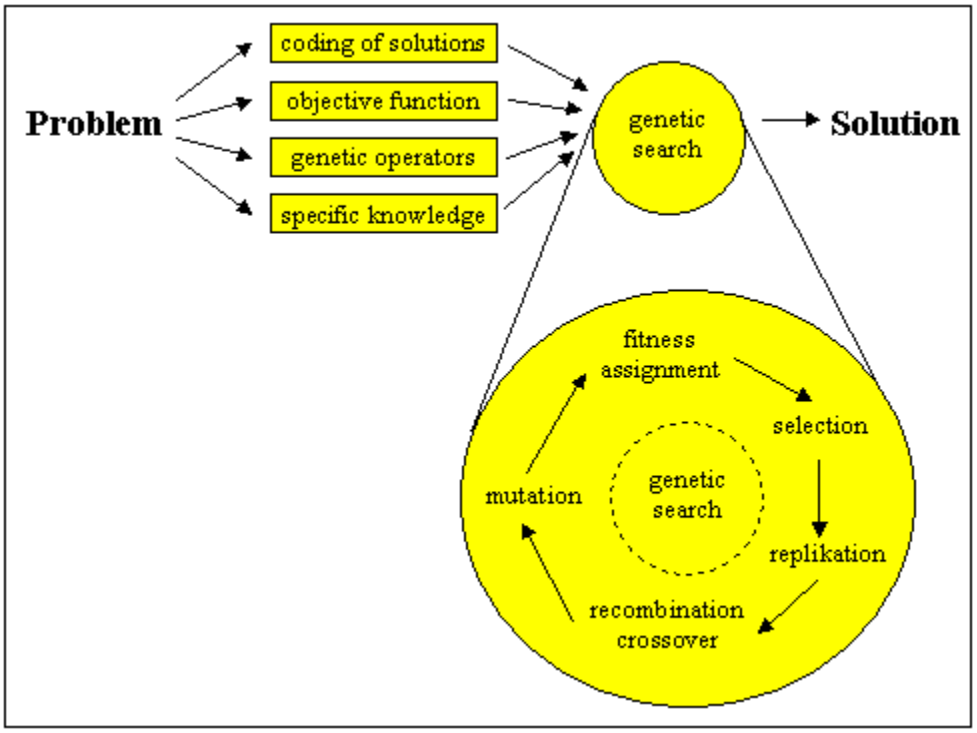


FIGURE 3.1 Problem solution using evolutionary algorithms

3.2 OVERVIEW

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Figure 2 shows the structure of a simple genetic algorithm. Evolutionary algorithms work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

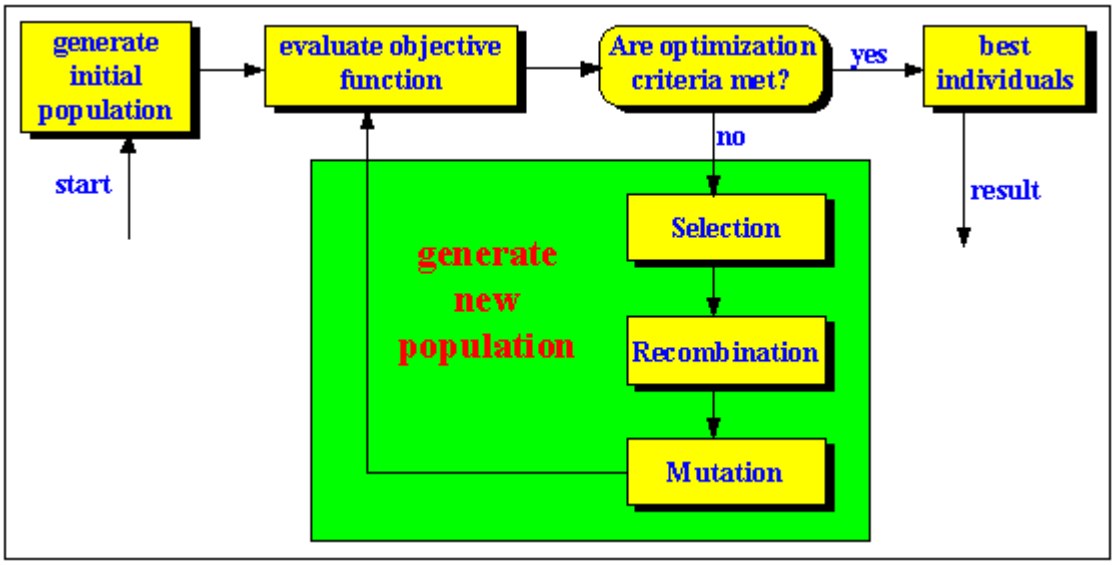


FIGURE 3.2 Structure of a single population evolutionary algorithm

Evolutionary algorithms differ substantially from more traditional search and optimization methods. A few of the most significant differences are:[14]

- Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- Evolutionary algorithms use probabilistic transition rules, not deterministic ones.

3.3 SELECTION

Selection determines, which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. In selection the individuals producing offspring are chosen. The first step is fitness assignment. Each individual in the selection pool receives a reproduction probability depending on the own objective value and the objective value of all other individuals in the selection pool. This fitness is used for the actual selection step afterwards.

The first step fitness assignment is by:

- Proportional fitness assignment or
- Rank-based fitness assignment

The actual selection is performed in the next step. Parents are selected according to their fitness by means of one of the following algorithms:

- Roulette-wheel selection
- Stochastic universal sampling
- Local selection
- Truncation selection or
- Tournament selection

3.3.1 Rank-Based Fitness Assignment

In rank-based fitness assignment, the population is sorted according to the objective values. The fitness assigned to each individual depends only on its position in the individuals rank and not on the actual objective value.

Rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. The reproductive range is limited, so that no individuals generate an excessive number of offspring. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure.

Rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment and, thus, is the method of choice.

Consider *Nind* the number of individuals in the population, *Pos* the position of an individual in this population (least fit individual has *Pos*=1, the fittest individual *Pos*=*Nind*) and *SP* the selective pressure. The fitness value for an individual is calculated as:

Linear ranking:

$$\text{Fitness}(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot (Pos - 1) / (Nind - 1)$$

Linear ranking allows values of selective pressure in [1.0, 2.0].

A new method for ranking using a non-linear distribution is introduced. The use of non-linear ranking permits higher selective pressures than the linear ranking method.

Non-linear ranking:

Fitness(*Pos*) = *Nind*·*X*^(*Pos* - 1) / sum(*X*^(*i* - 1)); *i* = 1:*Nind*

X is computed as the root of the polynomial:

0 = (*SP* - 1)·*X*^(*Nind* - 1) + *SP*·*X*^(*Nind* - 2) + ... + *SP*·*X* + *SP*

Non-linear ranking allows values of selective pressure in [1.0, *Nind* - 2.0].

Fig. 3.3 compares linear and non-linear ranking graphically.

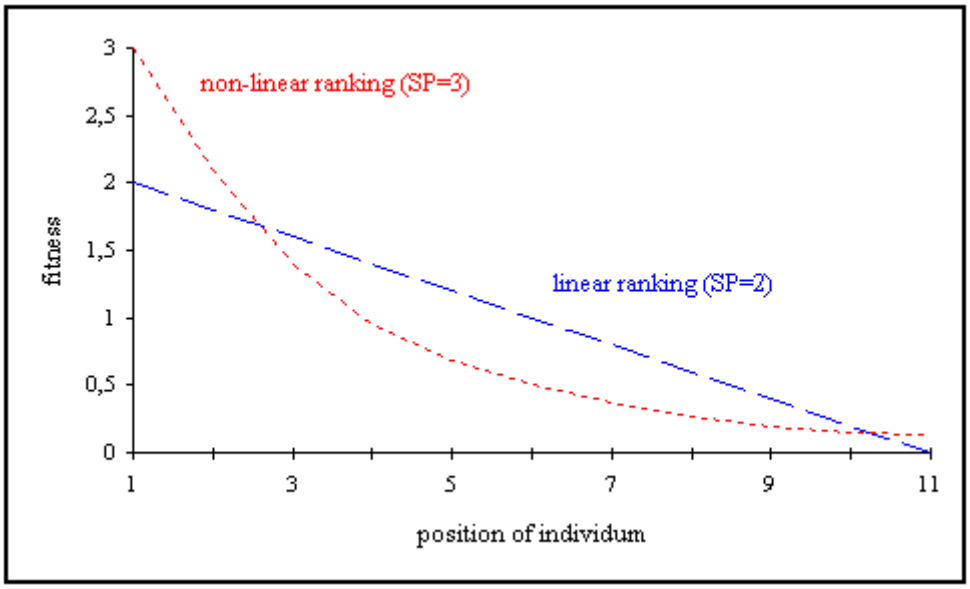


FIGURE 3.3 Fitness assignment for linear and non-linear ranking

The probability of each individual being selected for mating is its fitness normalized by the total fitness of the population.

Selection intensity:

$SelInt_{Rank}(SP) = (SP-1)\cdot(1/sqrt(pi))$.

Loss of diversity:

$LossDiv_{Rank}(SP) = (SP-1)/4$.

Selection variance:

$SelVar_{Rank}(SP) = 1-((SP-1)^2/pi) = 1-SelInt_{Rank}(SP)^2$.

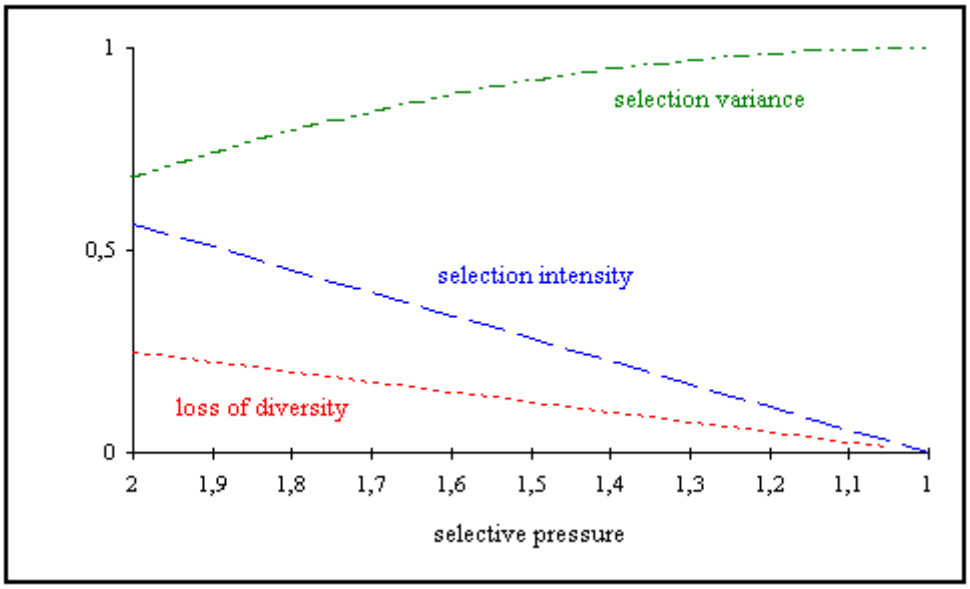


FIGURE 3.4 Properties of linear ranking

3.3.2 Roulette Wheel Selection

The simplest selection scheme is roulette-wheel selection, also called stochastic sampling with replacement. This is a stochastic algorithm and involves the following technique:
The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportional in size to the fitness, see Fig. 3.5.

Table 3.1 shows the selection probability for 11 individuals, linear ranking and selective pressure of 2 together with the fitness value. Individual 1 is the most fit individual and occupies the largest interval, whereas individual 10 as the second least fit individual has the smallest interval on the line (see figure 3.5). Individual 11, the least fit interval, has a fitness value of 0 and get no chance for reproduction

Number of individual	1	2	3	4	5	6	7	8	9	10	11

Fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
Selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

TABLE 3.1 Selection probability and fitness value

For selecting the mating population the appropriate number of uniformly distributed random numbers (uniform distributed between 0.0 and 1.0) is independently generated.

Sample of 6 random numbers:

0.81, 0.32, 0.96, 0.01, 0.65, 0.42.

Fig. 3.5 shows the selection process of the individuals for the example in table 3.1, together with the above sample trials.

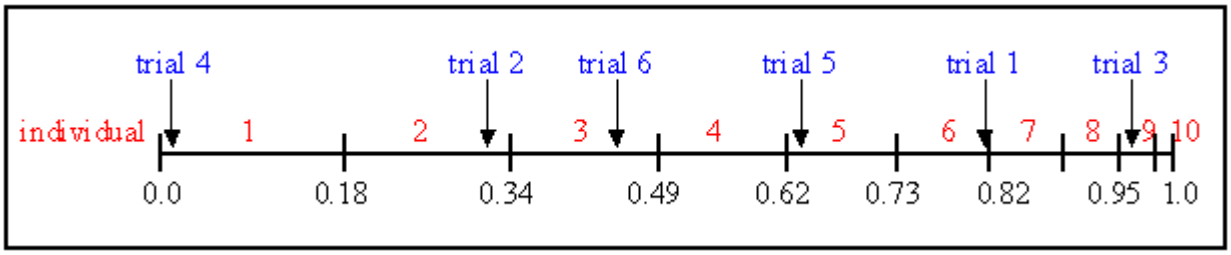


FIGURE 3.5 Roulette-wheel selection

After selection the mating population consists of the individuals:

1, 2, 3, 5, 6, 9.

The roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread.

3.3.3 Stochastic Universal Sampling

Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as many as there are individuals to be selected. Consider *Npointer*

the number of individuals to be selected, then the distance between the pointers are 1/*NPointer* and the position of the first pointer is given by a randomly generated number in the range [0, 1/*NPointer*].

For 6 individuals to be selected, the distance between the pointers is 1/6=0.167. Figure 6 shows the selection for the above example.

Sample of 1 random number in the range [0, 0.167]:

0.1.

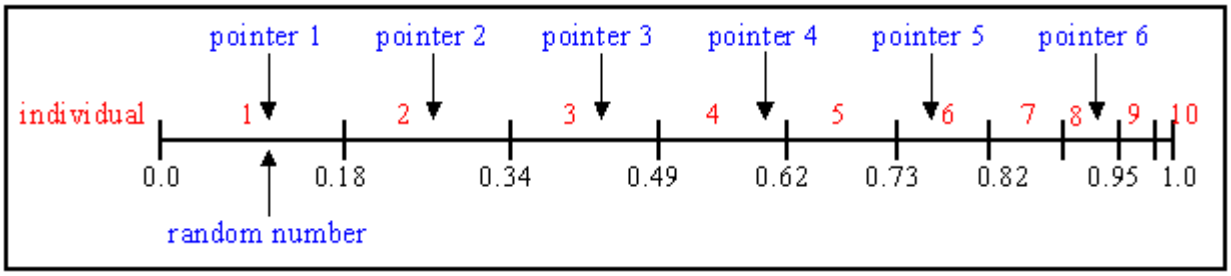


FIGURE 3.6 Stochastic universal sampling

After selection the mating population consists of the individuals:

1, 2, 3, 4, 6, 8.

Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.

3.3.4 Local Selection

In local selection every individual resides inside a constrained environment called the local neighborhood. (In the other selection methods the whole population or subpopulation is the selection pool or neighborhood.) Individuals interact only with individuals inside this region. The neighborhood is defined by the structure in which the population is distributed. The neighborhood can be seen as the group of potential mating partners.

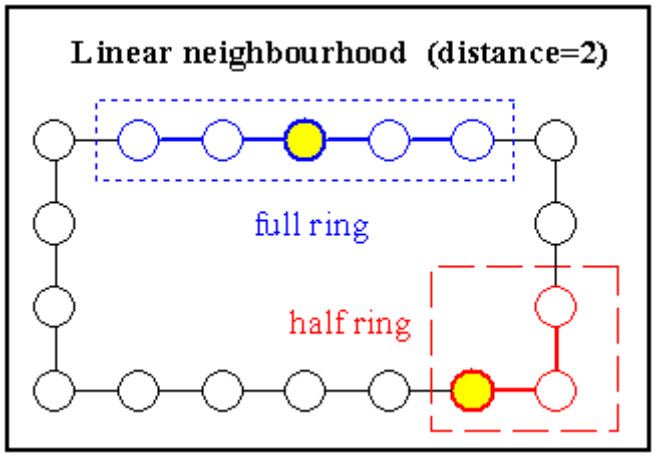


FIGURE 3.7 Linear neighborhood: full and half ring

The first step is the selection of the first half of the mating population uniform at random (or using one of the other mentioned selection algorithms, for example, stochastic universal sampling or truncation selection). Now a local neighborhood is defined for every selected individual. Inside this neighborhood the mating partner is selected (best, fitness proportional, or uniform at random).

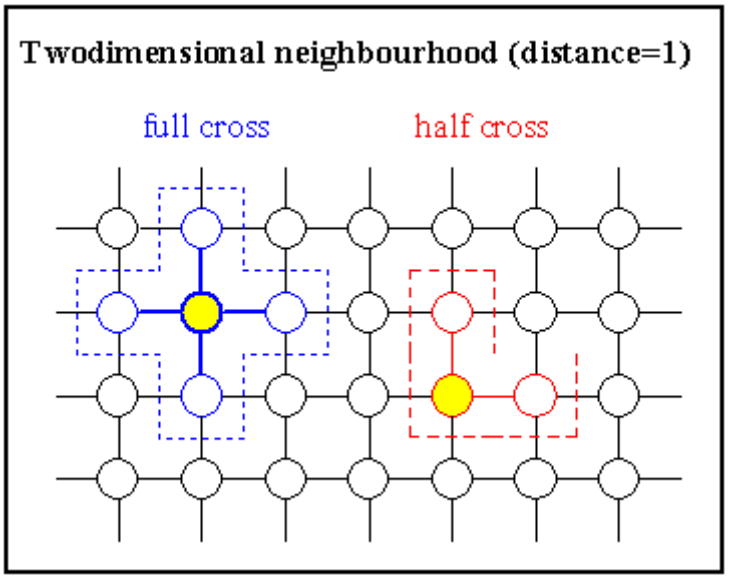


FIGURE 3.8 Two-dimensional neighborhood: full and half cross

The structure of the neighbourhood can be:

- Linear
    - Full ring, half ring (see Fig. 3. 7)
- Two-dimensional
    - Full cross, half cross (see Fig. 3. 8)
    - Full star, half star (see Fig. 3. 9)
- Three-dimensional and more complex with any combination of the above structures.

The distance between possible neighbors together with the structure determines the size of the neighborhood. Table 3.2 gives examples for the size of the neighborhood for the given structures and different distance values.

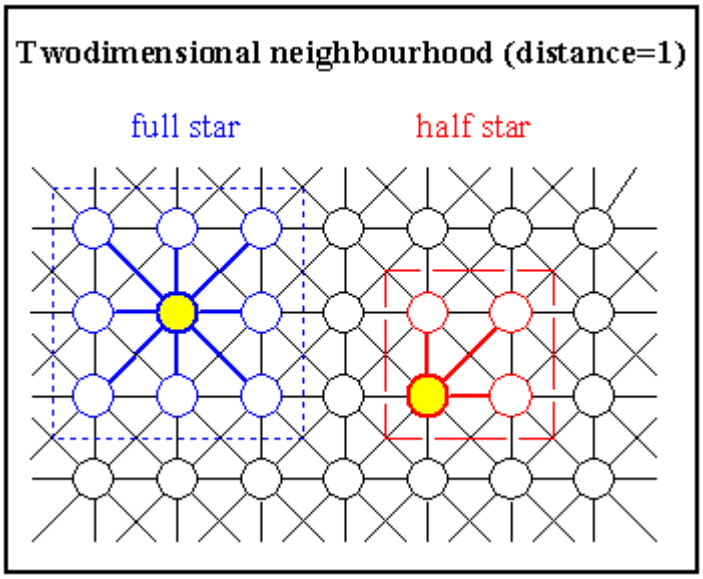


FIGURE 3.9 Two-dimensional neighborhood: full and half star

Between individuals of a population an 'isolation by distance' exists. The smaller the neighborhood, the bigger the isolation distance. However, because of overlapping neighborhoods, propagation of new variants takes place. This assures the exchange of information between all individuals.

structure	distance	
	1	2
full ring	2	4
half ring	1	2
full cross	4	8 (12)
half cross	2	4 (5)
full star	8	24
half star	3	8

TABLE 3.2 Number of neighbors for local selection

The size of the neighborhood determines the speed of propagation of information between the individuals of a population, thus deciding between rapid propagation or maintenance of a high diversity/variability in the population. A higher variability is often desired, thus preventing problems such as premature convergence to a local minimum. Local selection in a small neighborhood performed better than local selection in a bigger neighborhood. Nevertheless, the interconnection of the whole population must still be provided. Two-dimensional neighborhood with structure half star using a distance of 1 is recommended for local selection. However, if the population is bigger (>100 individuals) a greater distance and/or another two-dimensional neighborhood should be used.

3.3.5 Truncation Selection

Compared to the previous selection methods modeling natural selection truncation selection is an artificial selection method. It is used by breeders for large populations/mass selection.

In truncation selection individuals are sorted according to their fitness. Only the best individuals are selected for parents. These selected parents produce uniform at random offspring. The parameter for truncation selection is the truncation threshold *Trunc*. *Trunc* indicates the proportion of the population to be selected as parents and takes values ranging from 50%-10%. Individuals below the truncation threshold do not produce

offspring. The term selection intensity is often used in truncation selection. Table 3.3 shows the relation between both.

Truncation threshold	1%	10%	20%	40%	50%	80%
Selection intensity	2.66	1.76	1.2	0.97	0.8	0.34

TABLE 3.3 Relation between truncation threshold and selection intensity

Selection intensity:

$SelInt_{Trunc}(Trunc) = 1/Trunc/sqrt(2 \cdot pi) \cdot exp(-(f\text{^}2)/2)$.

Loss of diversity:

$LossDiv_{Trunc}(Trunc) = 1-Trunc$.

Selection variance:

$SelVar_{Trunc}(Trunc) = 1-SelInt_{Trunc}(Trunc) \cdot (SelInt_{Trunc}(Trunc)-f_c)$.

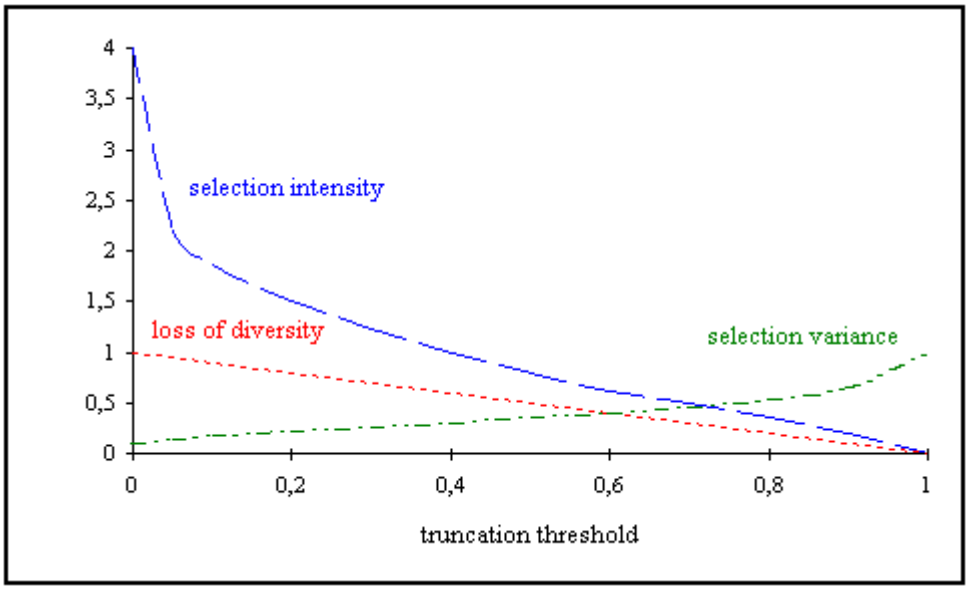


FIGURE 3.10 Properties of truncation selection

Comparison of selection schemes

Fraction of 1's in generation gen:

$$p(gen) = 0.5 \cdot (1+\sin(SelInt/\sqrt{n} \cdot gen + \arcsin(2 \cdot p_0 - 1)))$$

(n: dimension of objective function, p_0: fraction of 1's in initial random population) Convergence is characterized by p(GenConv) = 1. special case: $p_0 = 0.5$:

$$GenConv = \pi/2 \cdot \sqrt{n}/SelInt.$$

Number of generations to reach convergence:

Ranking selection:

$$GenConv_{Rank} = \sqrt{\pi \cdot n}/(2 \cdot (SP-1)).$$

Truncation selection:

$$GenConv_{Trunc} = \pi/2 \cdot \sqrt{n}/SelInt_{Trunc}.$$

Tournament selection:

$$GenConv_{Tour} = \pi/2 \cdot \sqrt{n/(2 \cdot (\log(Tour)-\log(\sqrt{4.14 \cdot \log(Tour)})))}.$$

The number of generations to reach convergence with a simple genetic algorithm is proportional to sqrt(n) and inversely proportional to selection intensity. (The population should be large enough to converge to the optimum and the initial population should be generated at random.) This would suggest a high selection intensity as best selection scheme. However, a high selection intensity leads to premature convergence and thus a poor quality of the solutions. The genetic algorithm works most effectively with the minimal population size N^*. This is the size where the population still converges to the optimum. N^* depends on the dimension of the objective function and the selection intensity.

As shown above the three selection methods behave similar assuming similar selection intensity. Figure 3.11 shows the relation between selection intensity and the appropriate parameter of the selection methods (selective pressure, truncation threshold and tournament size). It should be stated, that with tournament selection only discrete values can be assigned and linear ranking selection allows only a smaller range for the selection intensity.

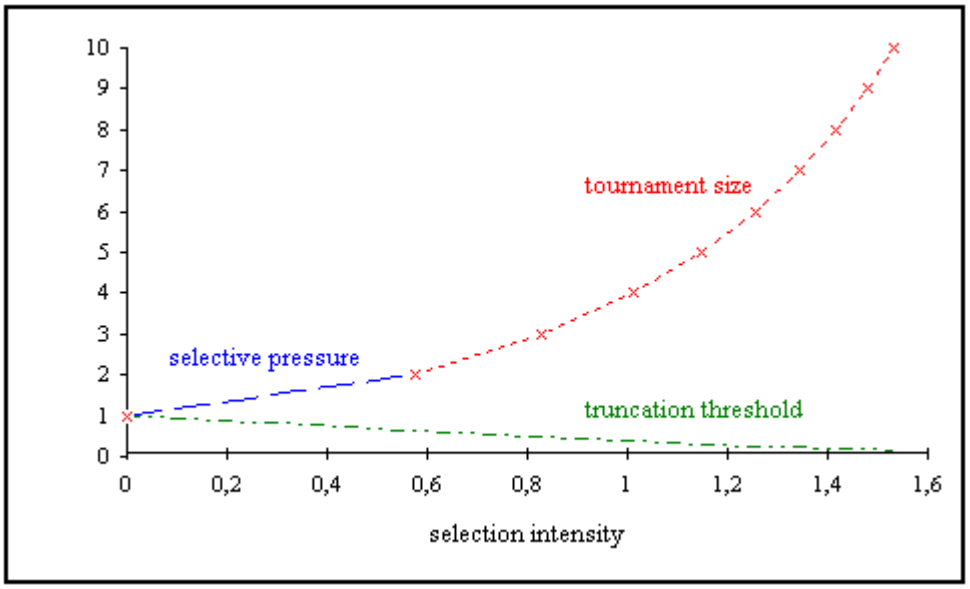


FIGURE 3.11 Dependence of selection parameter on selection intensity

However, the behavior of the selection methods is different. Thus, the selection methods will be compared on the parameters loss of diversity (Fig. 3.12) and selections variance (Fig. 3.13) on the selection intensity.

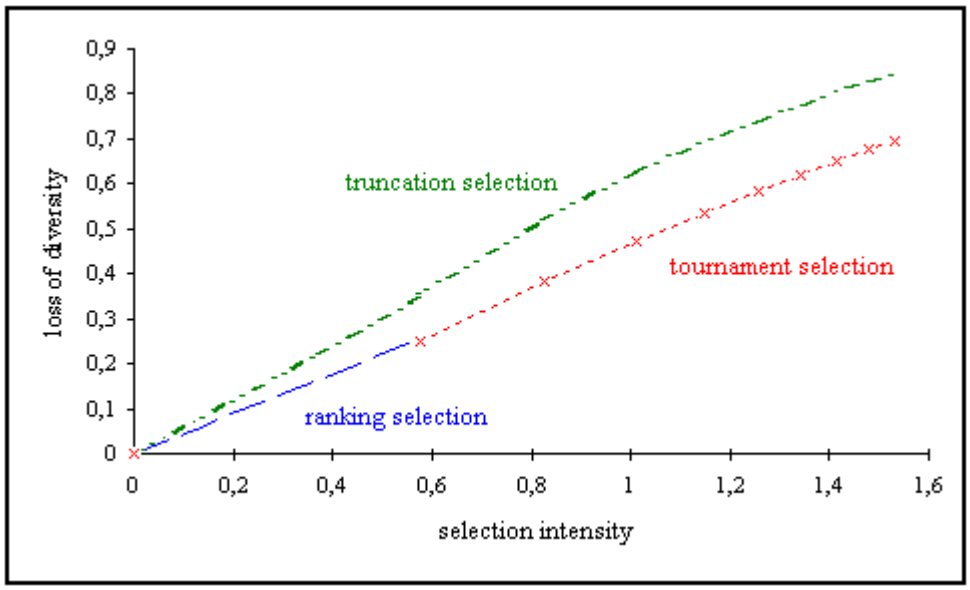


FIGURE 3.12 Dependence of loss of diversity on selection intensity

Truncation selection leads to a much higher loss of diversity for the same selection intensity compared to ranking and tournament selection. Truncation selection is more

likely to replace less fit individuals with fitter offspring, because all individuals below a certain fitness threshold don't have a probability to be selected. Ranking and tournament selection seem to behave similar. However, ranking selection works in an area where tournament selection doesn't work because of the discrete character of tournament selection.

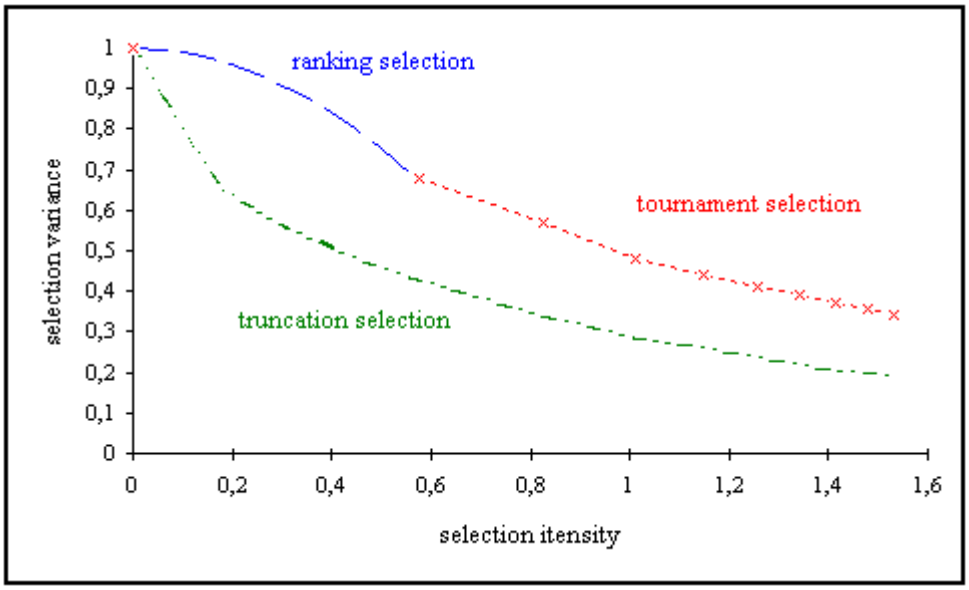


FIGURE 3.13: Dependence of selection variance on selection intensity

For the same selection intensity truncation selection leads to a much smaller selection variance than ranking or tournament selection. As can seen clearly ranking selection behaves similar to tournament selection. However, again ranking selection works in an area where tournament selection doesn't work because of the discrete character of tournament selection.

3.4 RECOMBINATION

3.4.1 Real Valued Recombination

3.4.1.1 Discrete recombination

Discrete recombination performs an exchange of variable values between the individuals. Consider the following two individuals with 3 variables each (3 dimensions), which will also be used to illustrate the other types of recombination:

individual 1 12 25 5

individual 2 123 4 34

For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability.

sample 1 2 2 1

sample 2 1 2 1

After recombination the new individuals are created:

offspring 1 123 4 5

offspring 2 12 4 5

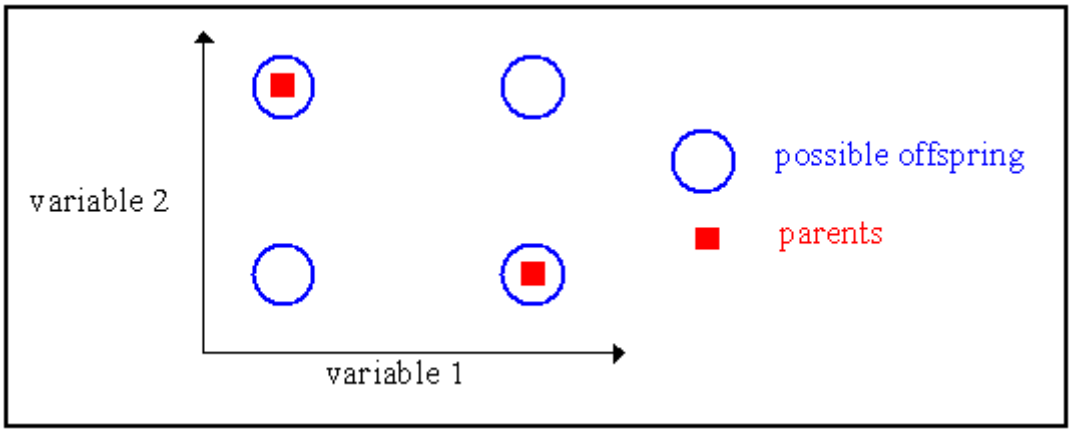


FIGURE 3.14 Possible positions of the offspring after discrete recombination

Discrete recombination generates corners of the hypercube defined by the parents. Fig. 3.14 shows the geometric effect of discrete recombination.

Discrete recombination can be used with any kind of variables (binary, real or symbols).

3.4.1.2 Intermediate Recombination

Intermediate recombination is a method only applicable to real variables (and not binary variables). Here the variable values of the offspring are chosen somewhere around and between the variable values of the parents.

Offspring are produced according to the rule:

offspring = parent 1 + Alpha (parent 2 - parent 1),

Where Alpha is a scaling factor chosen uniformly at random over an interval [$-d$, $1 + d$]. In intermediate recombination $d = 0$, for extended intermediate recombination $d > 0$. A good choice is d = 0.25. Each variable in the offspring is the result of combining the variables according to the above expression with a new Alpha chosen for **each** variable. See Fig. 3.15 for a picture of the area of the variable range of the offspring defined by the variables of the parents.

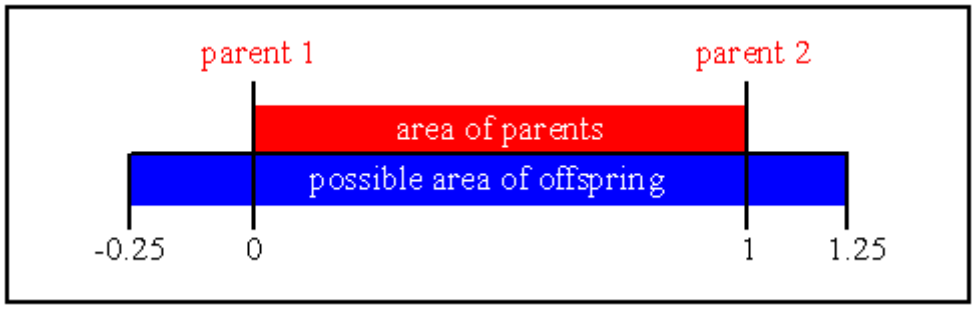


FIGURE 3.15 Area for variable value of offspring compared to parents in intermediate recombination

Consider the following two individuals with 3 variables each:

individual 1 12 25 5

individual 2 123 4 34

The chosen Alpha for this example are:

sample 1	0.5	1.1	-0.1
sample 2	0.1	0.8	0.5

The new individuals are calculated as:

offspring 1	67.5	1.9	2.1
offspring 2	23.1	8.2	19.5

Intermediate recombination is capable of producing any point within a hypercube slightly larger than that defined by the parents. Fig. 3.16 shows the possible area of offspring after intermediate recombination.

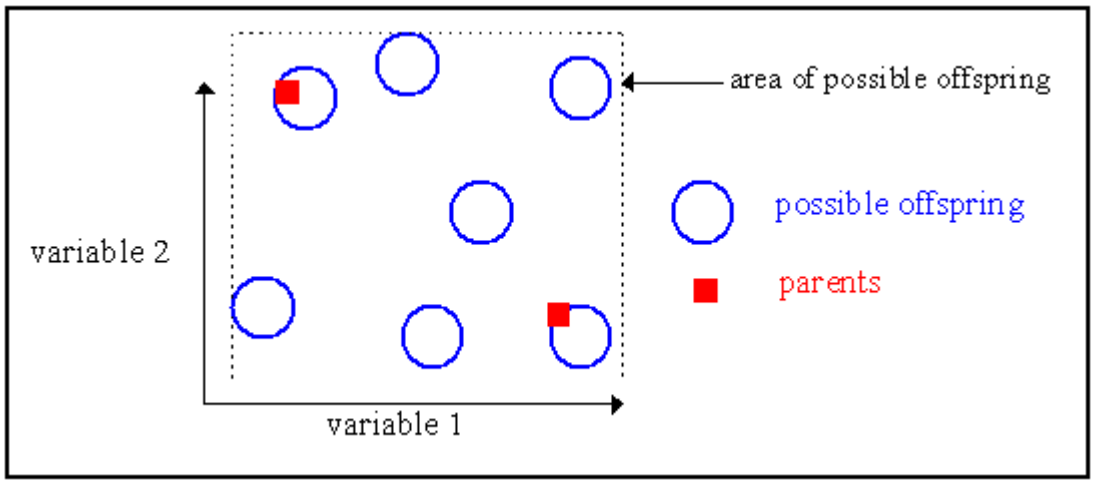


FIGURE 3.16 Possible area of the offspring after intermediate recombination

3.4.1.3 Line Recombination

Line recombination is similar to intermediate recombination, except that only one value of Alpha for all variables is used:

individual 1	12	25	5
individual 2	123	4	34

The chosen Alpha for this example are:

sample 1	0.5
sample 2	0.1

The new individuals are calculated as:

offspring 1	67.5	14.5	19.5
offspring 2	23.1	22.9	7.9

Line recombination can generate any point on the line defined by the parents. Fig. 3.17 shows the possible positions of the offspring after line recombination.

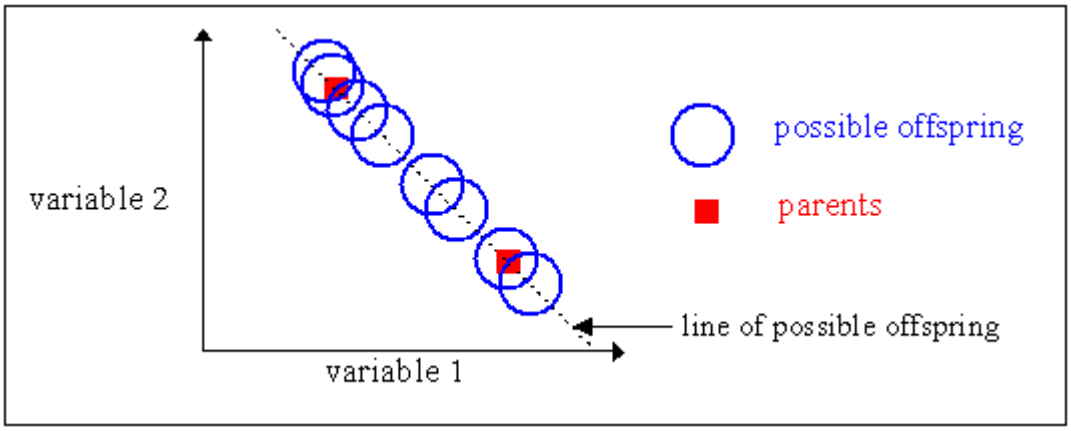


FIGURE 3.17 Possible positions of the offspring after line recombination

3.4.1.4 Extended Line Recombination

Extended line recombination generates offspring in a direction defined by the parents (line recombination). It tests more often outside the area defined by the parents and in the direction of parent 1. The point for the offspring is defined by features of the mutation operator of the Breeder Genetic Algorithm (Real valued mutation). The probability of small step sizes is greater than that of bigger steps (see figure). Extended line recombination is only applicable to real variables (and not binary or integer variables).

Offspring are produced according to the following rule:

- offspring 1 = parent 1 + *RecMx*·*range*·*delta*·*diff*, offspring 2 = parent 2 + *RecMx*·*range*·*delta*·(-*diff*).
- *RecMx* = 1 (- with probability 0.9),
- *range* = 0.5·domain of variable (search interval),
- *delta* = sum(*a*(*i*)· 2^-*i*), *a*(*i*) = 1 with probability 1/*m*, else *a*(*i*) = 0; *m* = 20; *i*=0:(m-1),
- *diff* = (parent 1 - parent 2)/parent 1 - parent 2

Consider the following two individuals with 3 variables each:

individual 1	12	25	5
individual 2	123	4	34

The chosen variables for this example are:

RecMx	-1		
range	50	50	50
delta	0.015625		
diff	-0.95	0.18	-0.25

The new individuals are calculated as:

offspring 1	12.7	24.8	5.19
offspring 2	11.2	25.1	4.80

3.4.2 Binary Valued Recombination (Crossover)

3.4.2.1 Single-Point Crossover

In single-point crossover one crossover position *k[1,2,...,Nvar*-1], *Nvar*: number of variables of an individual, is selected uniformly at random and the variables exchanged between the individuals about this point, then two new offspring are produced. Fig. 3.18 illustrates this process.

Consider the following two individuals with 11 binary variables each:

individual 1	0	1	1	1	0	0	1	1	0	1	0
individual 2	1	0	1	0	1	1	0	0	1	0	1

The chosen crossover position is:

crossover position 5

After crossover the new individuals are created:

offspring 1 0 1 1 1 0| 1 0 0 1 0 1

offspring 2 1 0 1 0 1| 0 1 1 0 1 0

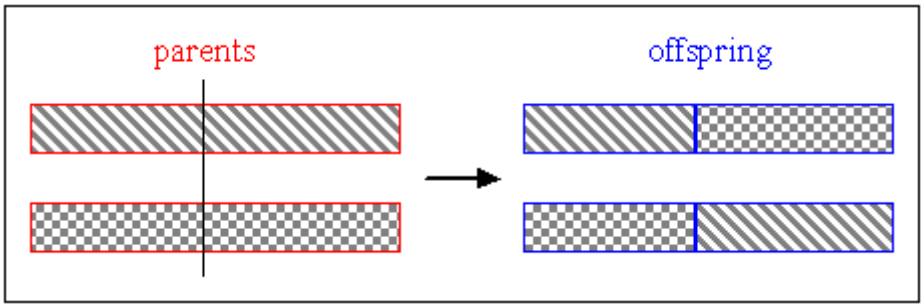


FIGURE 3.18 Single-point crossover

3.4.2.2 Multi-Point Crossover

For multi-point crossover, *m* crossover positions $k_i[1,2,...,Nvar-1]$, $i=1:m$, *Nvar*: number of variables of an individual, are chosen at random with no duplicates and sorted in ascending order. Then, the variables between successive crossover points are exchanged between the two parents to produce two new offspring. The section between the first variable and the first crossover point is not exchanged between individuals. Fig. 3.19 illustrates this process.

Consider the following two individuals with 11 binary variables each:

individual 1 0 1 1 1 0 0 1 1 0 1 0

individual 2 1 0 1 0 1 1 0 0 1 0 1

The chosen crossover positions are:

cross pos. (m=3) 2 6 10

After crossover the new individuals are created:

offspring 1 0 1| 1 0 1 1| 0 1 1 1| 1

offspring 2 1 0| 1 1 0 0| 0 0 1 0| 0

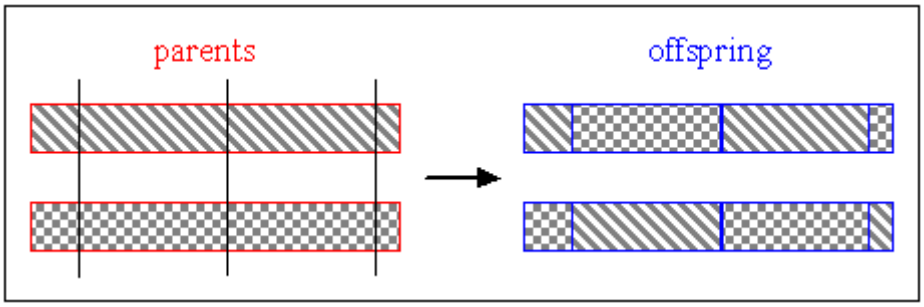


FIGURE 3.19 Multi-point crossover

The idea behind multi-point, and indeed many of the variations on the crossover operator, is that parts of the chromosome representation that contribute to the most to the performance of a particular individual may not necessarily be contained in adjacent substrings. Further, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favouring the convergence to highly fit individuals early in the search, thus making the search more robust.

3.4.2.3 Uniform Crossover

Single and multi-point crossover defines cross points as places between loci where an individual can be split. Uniform crossover generalizes this scheme to make every locus a potential crossover point. A crossover mask, the same length as the individual structure is created at random and the parity of the bits in the mask indicates which parent will supply the offspring with which bits.

Consider the following two individuals with 11 binary variables each:

individual 1 0 1 1 1 0 0 1 1 0 1 0

individual 2 1 0 1 0 1 1 0 0 1 0 1

For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability. Here, the offspring 1 is produced by taking the bit from parent 1 if the corresponding mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0. Offspring 2 is created using the inverse of the mask, usually.

sample 1	0 1 1 0 0 0 1 1 0 1 0
sample 2	1 0 0 1 1 1 0 0 1 0 1

After crossover the new individuals are created:

offspring 1	1 1 1 0 1 1 1 1 1 1 1
offspring 2	0 0 1 1 0 0 0 0 0 0 0

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation used and the particular coding for a given parameter set. This helps to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of the individual bits in the individuals representation. Spears and De Jong demonstrated how uniform crossover may be parameterized by applying a probability to the swapping of bits. This extra parameter can be used to control the amount of disruption during recombination without introducing a bias towards the length of the representation used.
The algorithm of uniform crossover is identical to discrete recombination.

3.4.2.4 Shuffle Crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

3.4.2.5 Crossover With Reduced Surrogate

The reduced surrogate operator constrains crossover to always produce new individuals wherever possible. This is implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.

3.5 MUTATION

After recombination offspring undergo mutation. Offspring variables are mutated by the addition of small random values (size of the mutation step), with low probability. The probability of mutating a variable is set to be inversely proportional to the number of variables (dimensions). The more dimensions one individual has as smaller is the mutation probability. Different papers reported results for the optimal mutation rate. Writes, that a mutation rate of 1/*n* produced good results for a broad class of test function. However, the mutation rate was independent of the size of the population. Similar results are reported in. For unimodal functions a mutation rate of 1/*n* was the best choice. An increase of the mutation rate at the beginning connected with a decrease of the mutation rate to 1/*n* at the end gave only an insignificant acceleration of the search. However, for multimodal functions a self-adaptation of the mutation rate could be useful.

3.5.1 Real Valued Mutation

Fig. 3.20 shows possible mutations for a real valued individual in two dimensions.

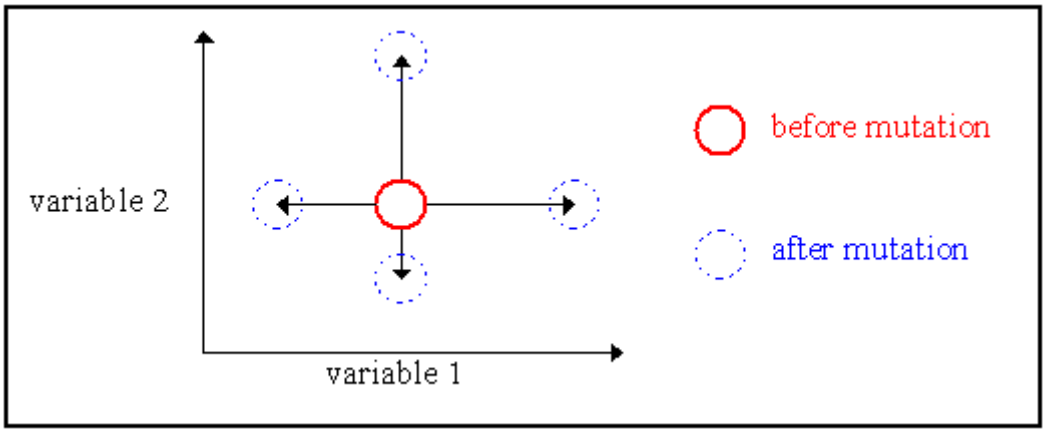


FIGURE 3.20 Effect of mutation

The size of the mutation step is usually difficult to choose. The optimal step size depends on the problem considered and may even vary during the optimization process. Small steps are often successful, but sometimes bigger steps are quicker.

The mutation operator of the Breeder Genetic Algorithm as proposed is:

- mutated variable = variable ± *range*·*delta*; (+ or - with equal probability)
- *range* = 0.5·domain of variable; (search interval),
- *delta* = sum($a(i)$ 2^-i), $a(i) = 1$ with probability $1/m$, else $a(i) = 0$; $m = 20$.

This mutation algorithm is able to generate most points in the hypercube defined by the variables of the individual and range of the mutation. However, it tests more often near the variable, that is, the probability of small step sizes is greater than that of bigger steps (see Fig 3.21). With m=20, the mutation algorithm is able to locate the optimum up to a precision of (*range*·2^-19).

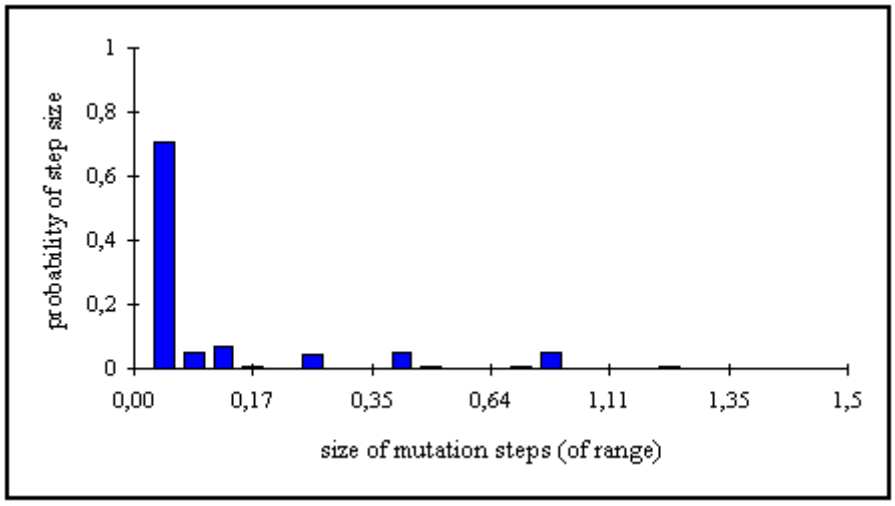


FIGURE 3.21 Probability and size of mutation steps (compared to range)

3.5.2 Binary Mutation

For binary valued individuals mutation means flipping of variable values. For every individual the variable value to change is chosen uniform at random. Table 3.4 shows an example of a binary mutation for an individual with 11 variables, variable 4 is mutated.

Before mutation	0	1	1	1	0	0	1	1	0	1	0
After mutation	0	1	1	0	0	0	1	1	0	1	0

TABLE 3.4 Individual before and after binary mutation

Assuming that the above individual decodes a real number in the bounds [1, 10], the effect of the mutation depends on the actual coding. Table 3.5 shows the different numbers of the individual before and after mutation for binary/gray and arithmetic/logarithmic coding.

Arithmetic		Logarithmic	
Binary	Gray	Binary	Gray
5.0537	4.2887	2.8211	2.3196
4.4910	3.3346	2.4428	1.8172

TABLE 3.5 ***Result of the binary mutation***

3.6 REINSERTION

Once the offspring have been produced by selection, recombination and mutation of individuals from the old population, the fitness of the offspring may be determined. If less offspring are produced than the size of the original population then to maintain the size of the original population, the offspring have to be reinserted into the old population. Similarly, if not all offspring are to be used at each generation or if more offspring are generated than the size of the old population then a reinsertion scheme must be used to determine which individuals are to exist in the new population.

The used selection method determines the reinsertion scheme: local reinsertion for local selection and global reinsertion for all other selection methods.

3.6.1 Global Reinsertion

Different schemes of global reinsertion exist:

- Produce as many offspring as parents and replace all parents by the offspring (pure reinsertion).
- Produce less offspring than parents and replace parents uniformly at random (uniform reinsertion).
- Produce less offspring than parents and replace the worst parents (elitist reinsertion).
- Produce more offspring than needed for reinsertion and reinsert only the best offspring (fitness-based reinsertion).

Pure Reinsertion is the simplest reinsertion scheme. Every individual lives one generation only. This scheme is used in the simple genetic algorithm. However, it is very likely, that very good individuals are replaced without producing better offspring and thus, good information is lost.

The elitist combined with fitness-based reinsertion prevents this losing of information and is the recommended method. At each generation, a given number of the least fit parents are replaced by the same number of the most fit offspring (see Fig. 3.22). The fitness-based reinsertion scheme implements a truncation selection between offspring before inserting them into the population (i.e. before they can participate in the reproduction process). On the other hand the best individuals can live many generations. However, every generation some new individuals are inserted. It is not checked whether the parents are replaced by better or worse offspring.

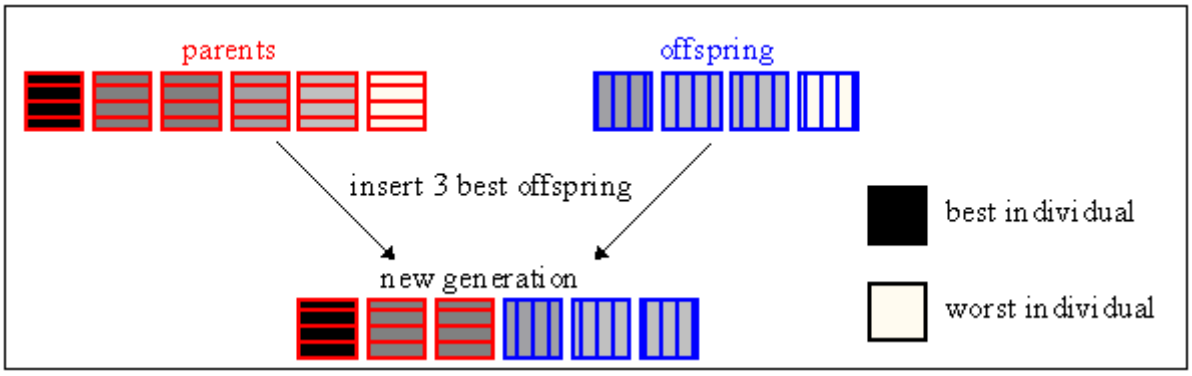


FIGURE 3.22 Scheme for elitist insertion

Because parents may be replaced by offspring with a lower fitness, the average fitness of the population can decrease. However, if the inserted offspring are extremely bad, they will be replaced with new offspring in the next generation.

3.6.2 Local Reinsertion

In local selection individuals are selected in a bounded neighborhood. The reinsertion of offspring takes place in exactly the same neighborhood. Thus, the locality of the information is preserved.

The used neighborhood structures are the same as in local selection. The parent of an individual is the first selected parent in this neighborhood.

For the selection of parents to be replaced and for selection of offspring to reinsert the following schemes are possible:

- Insert every offspring and replace individuals in neighborhood uniform at random,
- Insert every offspring and replace weakest individuals in neighborhood,
- Insert offspring fitter than weakest individual in neighborhood and replace weakest individuals in neighborhood,
- Insert offspring fitter than weakest individual in neighborhood and replace parent,
- Insert offspring fitter than weakest individual in neighborhood and replace individuals in neighborhood uniform at random,
- Insert offspring fitter than parent and replace parent.

3.7 PARALLEL IMPLEMENTATIONS

3.7.1 Migration

The migration model divides the population in multiple subpopulations. These subpopulations evolve independently from each other for a certain number of generations (isolation time). After the isolation time a number of individuals is distributed between the subpopulations (migration). The number of exchanged individuals (migration rate), the selection method of the individuals for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulations and the exchange of information between subpopulations.

The parallel implementation of the migration model showed not only a speedup in computation time, but it also needed less objective function evaluations when compared to a single population algorithm. So, even for a single processor computer, implementing the parallel algorithm in a serial manner (pseudo-parallel) delivers better results (the algorithm finds the global optimum more often or with less function evaluations).

The selection of the individuals for migration can take place:

- Uniform at random (pick individuals for migration in a random manner),
- Fitness-based (select the best individuals for migration).

Many possibilities exist for the structure of the migration of individuals between subpopulations. For example, migration may take place:

- Between all subpopulations (complete net topology - unrestricted), see Fig. 3.23,
- In a ring topology, see Fig. 3.25,
- In a neighborhood topology, see Fig. 3.26

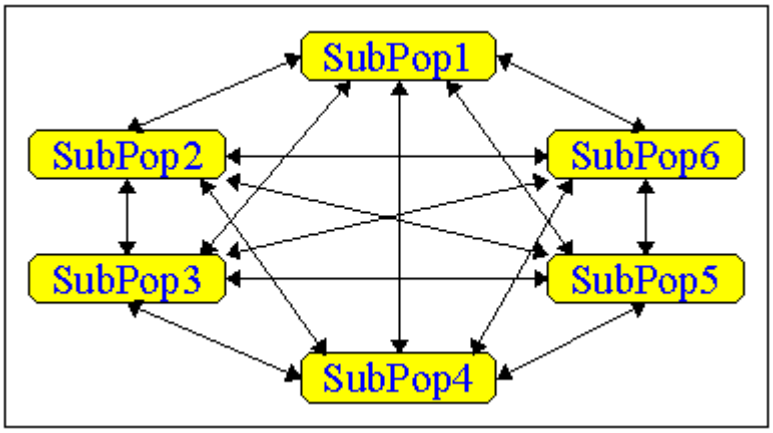


FIGURE 3.23 Unrestricted migration topology (Complete net topology)

The most general migration strategy is that of unrestricted migration (complete net topology). Here, individuals may migrate from any subpopulation to another. For each subpopulation, a pool of potential immigrants is constructed from the other subpopulations. The individual migrants are then uniformly at random determined from this pool.

Fig. 3.24 gives a detailed description for the unrestricted migration scheme for 4 subpopulations with fitness-based selection. Subpopulations 2, 3 and 4 construct a pool of their best individuals (fitness-based migration). 1 individual is uniformly at random chosen from this pool and replaces the worst individual in subpopulation 1. This cycle is performed for every subpopulation. Thus, it is ensured that no subpopulation will receive individuals from itself.

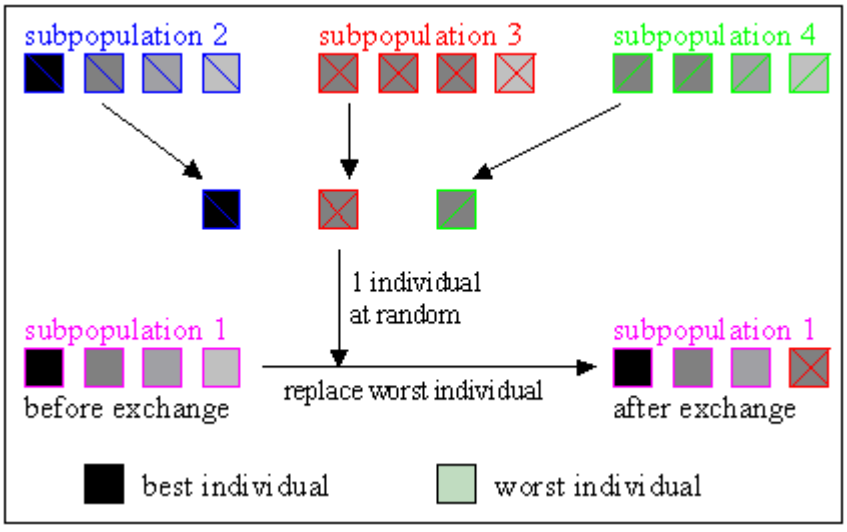


FIGURE 3.24 Scheme for migration of individuals between subpopulations

The most basic migration scheme is the ring topology. Here individuals are transferred between directionally adjacent subpopulations. For example, individuals from subpopulation 6 migrate only to subpopulation 1 and individuals from subpopulation 1 only migrate to subpopulation 2.

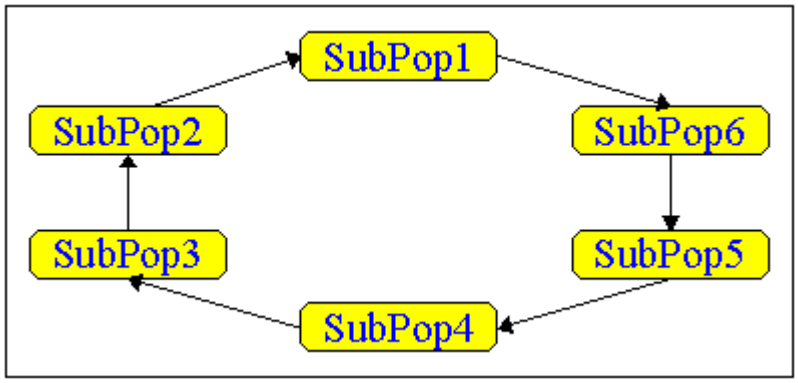


FIGURE 3.25 Ring migration topology

A similar strategy to the ring topology is the neighborhood migration of Fig. 3.26. Like the ring topology, migration is made only between nearest neighbors. However, migration may occur in either direction between subpopulations. For each subpopulation, the possible immigrants are determined, according to the desired selection method, from adjacent subpopulations and a final selection is made from this pool of individuals (similar to Fig. 3.25).

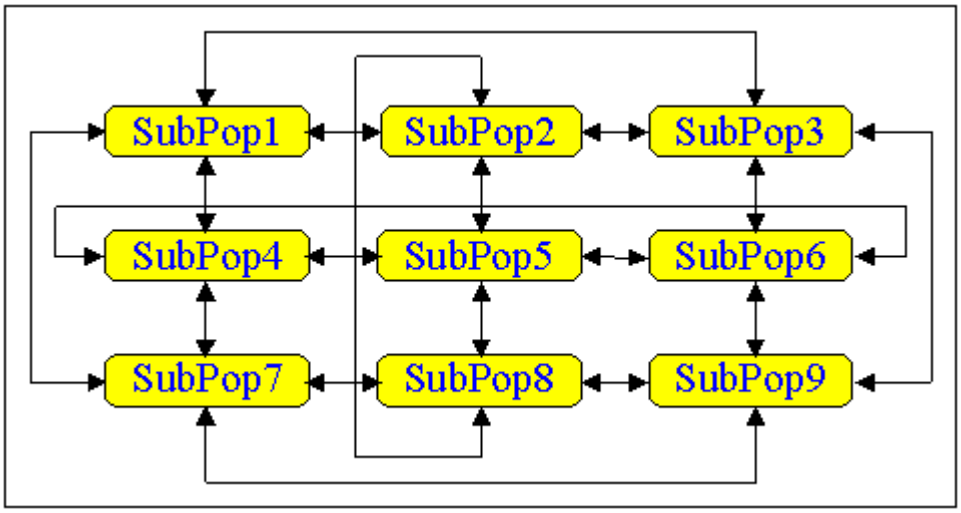


FIGURE 3.26: Neighborhood migration topology (2-D implementation)

Fig3.26 shows a possible scheme for a 2-D implementation of the neighborhood topology. Sometimes this structure is called a torus.

With the Multipopulation genetic algorithm, for every function tested better results were obtained than for a single population algorithm with proportionally more individuals.

7.2 Global model - worker/farmer

The global model employs the inherent parallelism of genetic algorithms (population of individuals). The Worker/Farmer algorithm is a possible implementation.

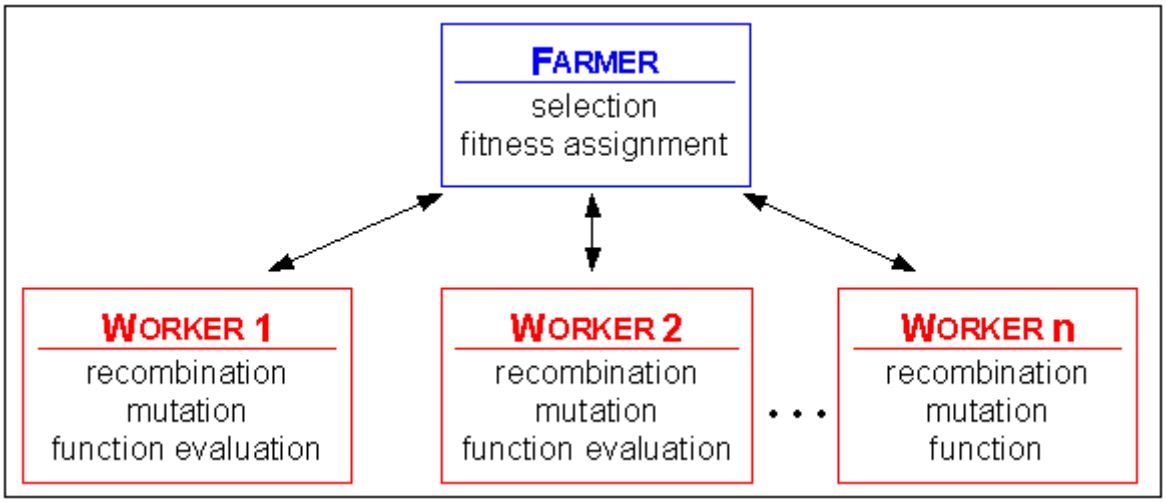


FIGURE 3.27 Worker/Farmer genetic algorithm

3.7.2 Diffusion Model

The diffusion model handles every individual separately and selects the mating partner in a local neighborhood similar to local selection. Thus, a diffusion of information through the population takes place. During the search virtual islands, see Fig. 3.28 will evolve.

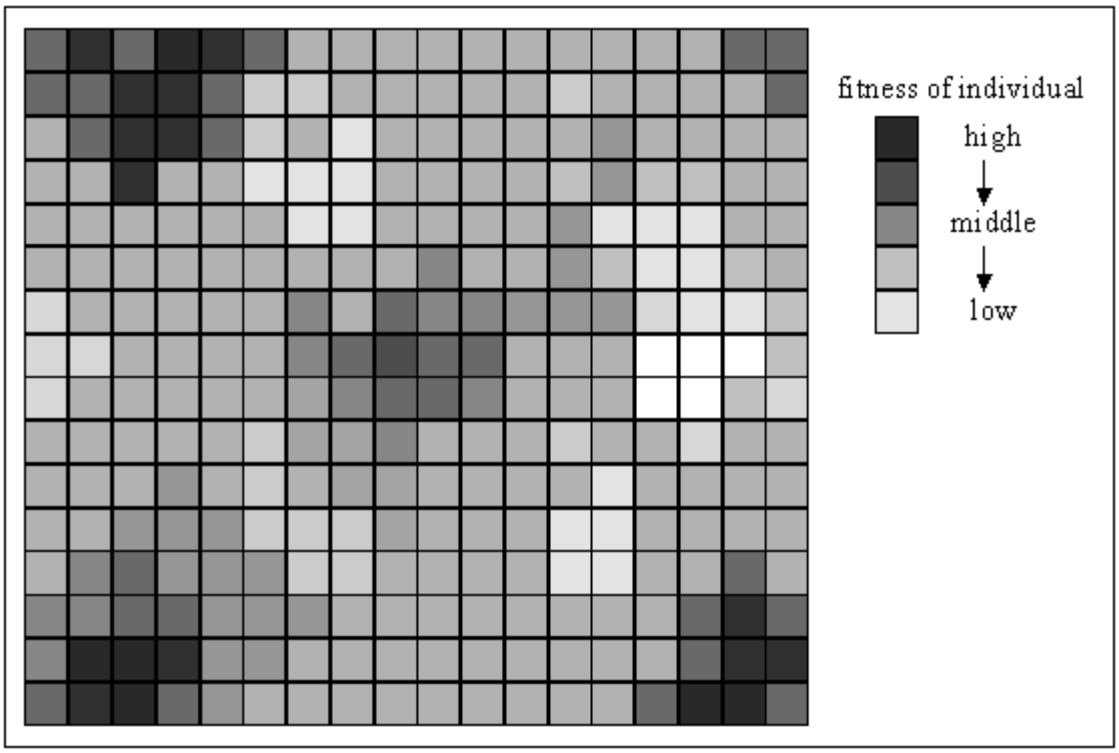


FIGURE 3.28: Diffusion genetic algorithm

CHAPTER 4

CHARACTER RECOGNITION

The aim of this dissertation is to recognize all the vowels and consonants of printed English characters using Genetic Algorithm. The additional features included in the project are:

- Multiple character recognition
- Rotational invariance

The recognition process uses two-stage classification approach, which is a hybrid of structural technique and genetic algorithm. For recognition, the character image is subjected to an algorithm for finding out the primitive or certain structural features of characters. These features are then applied in fitness function that is used by genetic algorithm to find the character in the image. In the following sections, the various stages involved in the recognition process are described.

4.1 STEPS FOR CHARACTER RECOGNITION

English is the most widely used language all over the world. It consists of 26 total characters, which include 5 vowels and 21 consonants. The list of all the alphabets is given below:

The steps to perform single character recognition are shown in Fig. 4.1. All the vowels and consonants are included during recognition.

The stages of single character recognition include:

1. Extracting character matrix pixels from the image.
2. Extraction of features of the character.

3. Developing a genetic algorithm tool.[13]
4. Developing a fitness function for the genetic algorithm tool based on the features extracted.
5. Determination of the character with the best recognition accuracy.

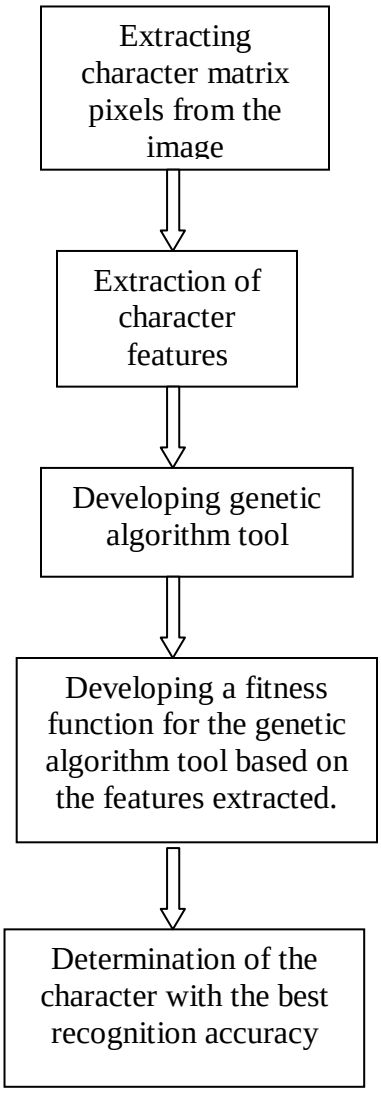


FIGURE 4.1 Steps for Character Recognition

These steps for achieving character recognition and additional features are discussed in detail in the following sections.

4.2 EXTRACTING CHARACTER MATRIX PIXELS FROM THE IMAGE

For recognition purpose 26 characters are taken into consideration. In order to locate the data (in our case character) position in the image, first step is to clip the background and obtain the pixels that contain only data. This not only helps in locating the data pixels in the whole image but also reduces the size of the data file on which we have to perform further computations. For this purpose first the RGB format of the image is converted into binary image format (first to gray image and then finally to binary format). This will convert the M x N x 3 image format into M x N format, where, M X N is the size of the image. Image can be in any of the formats compatible with MATLAB tool i.e. bmp, tiff, jpeg, png, ico, ras, pmp, hdf, xwd etc. But here '.bmp' images is used. Now once the image is converted to binary image, we get a matrix in the form of 1's and 0's. As the image will have data points in the form of 0's (black), and background pixels as 1's (white), we can separate out our character from the background.

This step is shown in detail by the following figures:

An arbitrary input data image carrying the character to be recognized is shown fig. 4.2:

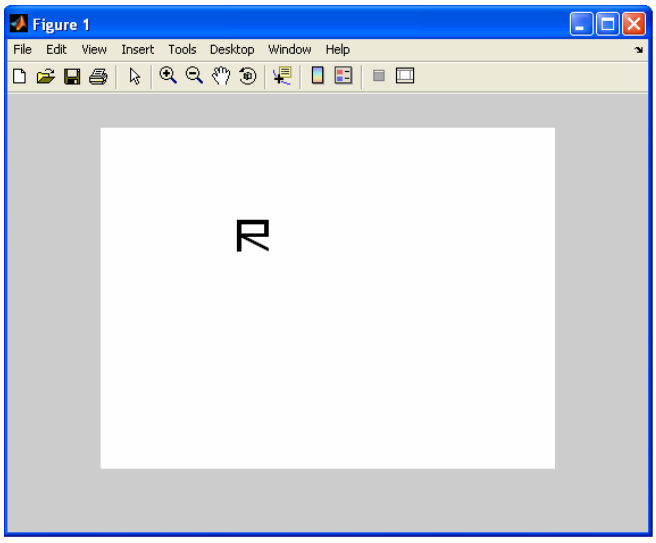


FIGURE 4.2 Input data image

The image left after extraction of the data character is shown in Fig. 4.3:

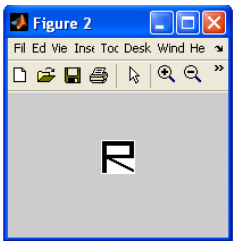


FIGURE 4.3 Image with extracted data character

4.3 EXTRACTION OF FEATURES OF THE CHARACTER

After the character pixels are segmented from the image, next step is to represent it in a form suitable for the further processing. Representation of a region involves two choices:

1. In terms of its external characteristics (its boundary)
2. In terms of its internal characteristics (the pixels comprising the region)

An external representation is chosen when the primary focus is on shape characteristics while an internal representation is selected when the primary focus is on regional properties, such as color and texture. Characters can be recognized depending on their shape. So features chosen are in the form of their external characteristics like shape number, number of horizontal, vertical or slant lines etc.

4.3.1 Shape Number

To get the shape number of a particular character, its Freeman Chain Code is detected. Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specific length and direction. Typically this representation is based on 4- or 8-connectivity of the segments. A pixel ‘p’ at coordinates (x, y) has four horizontal and vertical neighbors whose coordinates are given by:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

This set of pixels, called the 4-neighbors of p, is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y), and some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image.

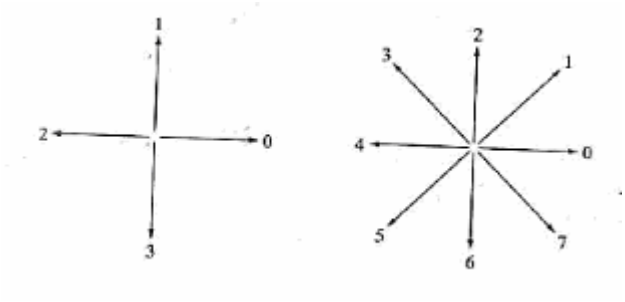
The four diagonal neighbors of p have coordinates:

(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)

and are denoted by $N_D(p)$. These points, together with the 4-neighbors, are called the 8-neighbors of p, denoted by $N_8(p)$.

To establish if two pixels are *connected*, it must be determined if they are neighbors and their gray levels satisfy a specified criterion of similarity (for e.g., they have same values).

The direction of each segment of connected sequence of straight-line, in chain codes, is coded by using a numbering scheme such as the ones shown in Fig. 4.3



(a) (b)

FIGURE 4.3 Direction numbers for (a) 4 directional chain code, (b) 8 directional chain code

Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y- directions, so a chain code could be generated by following a boundary in , say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels. This method is generally unacceptable for 2 principal reasons:

1. The resulting chain of codes tends to be quite long and
2. Any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the shape of the boundary.

An approach frequently used to circumvent the problems just discussed is to resample the boundary by selecting larger grid spacing, as illustrated in Figure 4.4a. Then, as the boundary is traversed, a boundary point is assigned to each node of the large grid, depending on the proximity of the original boundary to that node, as shown in Figure

4.4b. The re-sampled boundary obtained in this way then can be represented by a 4- or 8-code, as shown in Figure 4.4c & d, respectively. The starting point in figure 4.4c is (arbitrarily) at the top, left dot, and the boundary is the shortest allowable 4- or 8- path in the grid of Figure 4.4b. The boundary representation in Figure 4.4c is the chain code 0033….01, and in Figure 4.4d, it is the code 0766….12. As might be expected, the accuracy of the resulting code representation depends on the spacing of the sampling grid.

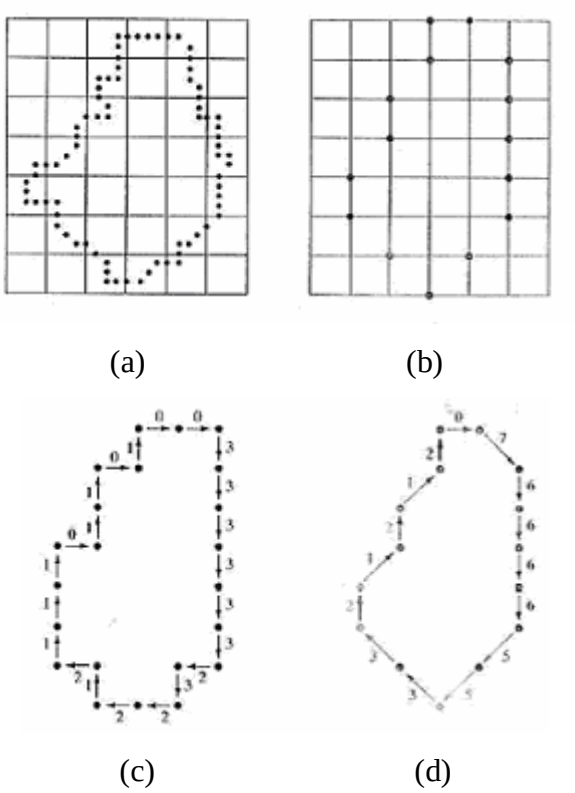


FIGURE 4.4 (a) Digital boundary with resampling grid superimposed. (b) Result of resampling. (c) 4-directional chain code. (d) 8-directional chain code.

The chain code of a boundary depends on the starting point. However, the code can be normalized with respect to the starting point by straight forward procedure: we simply treat the chain code as a circular sequence of direction numbers and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude. We can normalize also for the rotation by using the first difference of the chain code instead of the code itself. This difference is obtained by counting the numbers of

direction changes (in a counter-clockwise direction) that separate two adjacent elements of the code. For an instance, the first difference of the code direction of the 4- direction chain code 10103322 is 3133030. If we elect to treat the code as a circular sequence then the first element of the difference is computed by using the transition between the last and first components of the chain. Here, the result is 33133030. Size normalization can be achieved by altering the size of the re-sampling grid.[2]

The shape number follows from the first difference as shown in the Figure 4.5

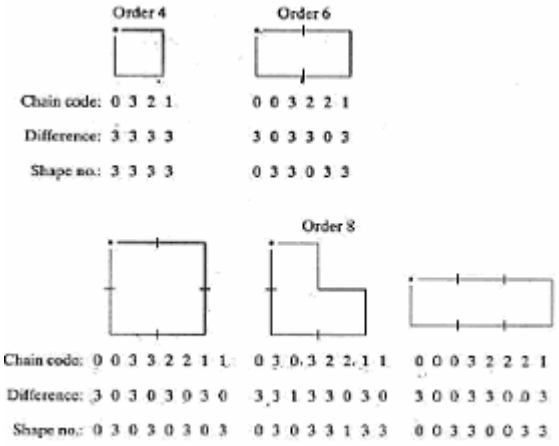
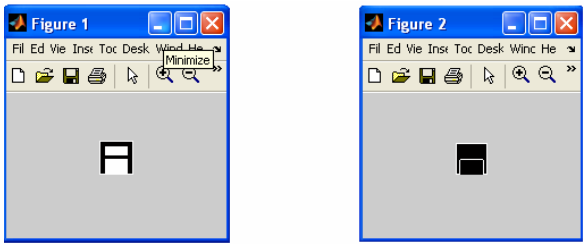


FIGURE 4.5 All shapes are from Fig. 4.3(a) and the dots indicate the starting point.

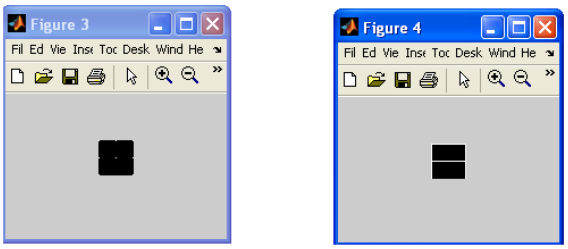
The original image obtained and its boundary shape are shown in the Fig. 4.6



(a) (b)

FIGURE 4.6 (a) Original image. (b) Boundary shape.

Its, sampled image and connected images are shown in Fig. 4.7



(a) (b)

FIGURE 4.7 (a) Sampled image. (b) Connected image.

Now the shape number of this image comes out to be:

```
>> shape_No

shape_No =

    0    6    0    4    2    0    2    4    0    6
```

4.3.2 Line Detection Algorithm

The next feature used is based on detection of number of lines of a particular orientation. To trace out such lines in any binary image, *convolution of masks in spatial domain* is carried out.

4.3.2.1 Convolution of Masks in Spatial Domain

Some neighborhood operations work with the values of the image pixels in the neighborhood and the corresponding values of a subimage that has the same dimensions as the neighborhood is called a *filter, mask, kernel, template or window*. The values in a filter subimage are referred to as *coefficients*, rather than pixels.

The concept of filtering has its root in the use of Fourier Transform for the signal processing in the so called *frequency domain*. But, filtering operations that are performed directly on the pixels of an image are termed as *spatial filtering* to differentiate this type of process from the more traditional frequency domain filtering.

The mechanics of spatial filtering are illustrated in Fig. 4.8(a). The process consists simply of moving the filter mask from point to point in an image. For *linear* spatial filtering, the response at each point (x,y) is given by a sum of products of the filter coefficients and the corresponding image pixels in the area scanned by the filter mask. For the 3 x 3 mask shown in Fig. 4.8(a), the result (or response), R, of linear filtering with the filter mask at a point of (x,y) in the image is

$$R = w(-1,-1)\, f(x-1, y-1) + w(-1,0)\, f(x-1, y) + \ldots\ldots + w(0,0)\, f(x, y) + \ldots\ldots + w(1,0)\, f(x+1, y) + w(1,1)\, f(x+1, y+1)$$

which is the sum of products of the mask coefficients with the corresponding pixels directly under the mask. In particular the coefficient w (0,0) coincides with image value f (x,y), indicating that the mask is centered at (x,y) when the computation of the sum of products takes place. For a mask of size m x n, we assume that m = 2a+1 and n = 2b+1, where a & b are non-negative integers.

In general, linear filtering of an image *f* of size M x N with a filter mask of size m x n is given by the Eq. 4.3-1

$$G(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t) \qquad (4.3\text{-}1)$$

Where, a = (m-1)/2 and b = (n-1)/2. To generate a complete filtered image, this equation must be applied for x = 0,1,2……M-1 and y = 0,1,2…..N-1. This assures that the mask processes all pixels in the image.

The process of linear filtering given in Eq. 4.3-1 is similar to a frequency domain concept *convolution*. For this reason, linear spatial filtering often is referred to as “convolving a mask with an image”. Similarly, filter masks are sometimes called *convolution mask*.[16]

When interest lies on the response, R, of an m x n mask at any point (x,y) and not on the mechanics of implementing mask convolution, the notation is simplified by using the following equation:

$$R = w_1 z_1 + w_2 z_2 + w_3 z_3 + \ldots\ldots\ldots\ldots + w_{mn} z_{mn}$$

$$= \sum_{i=1}^{mn} w_i z_i$$

where the *w*'s are the mask coefficients, the *z*'s are the values of the image gray levels corresponding to those coefficients, and *mn* is the total number of coefficients in the mask. For the 3 x 3 general mask shown in Fig.4.8 (b), the response at any point (x, y) in the image is given by:

$$R = w_1 z_1 + w_2 z_2 + w_3 z_3 + \ldots\ldots\ldots\ldots + w_9 z_9$$

$$= \sum_{i=1}^{9} w_i z_i$$

Consider the masks shown in Fig. 4.9. If the first mask were moved around in a image it would respond more strongly to lines (one pixel thick) oriented horizontally. With a constant background, the maximum response would result when the line past through the middle row of the mask. This is easily verified by sketching a simple array of 1's with a line of different gray levels (say, 5's) running horizontally through the array. A similar experiment would reveal that the second mask in Fig 4.6 responds best to lines oriented at + 45º; the third mask to vertical lines; and the fourth mask to lines in the –45º direction. These directions can be established also by noting that the preferred direction of each mask is weighted with a larger coefficient (i.e., 2) then other possible directions. Note that the coefficients in the each mask sum to zero indicating a zero response from the masks in areas of constant gray level.

Once a particular mask has been moved around an image, the number of lines of that particular orientation can be calculated in the image.

The results carried out with one of the mask's shown in Fig. 4.9 are:

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+45°

-1	2	-1
-1	2	-1
-1	2	-1

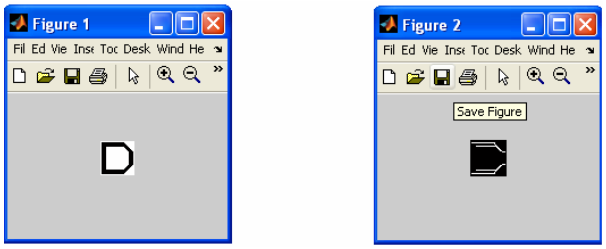
Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45°

FIGURE 4.9 Line Masks

The original image and the image after horizontal mask was moved around the image are shown in Fig. 4.10



(a) (b)

FIGURE 4.10 (a) Original image. (b) Image after horizontal mask was moved around

the image.

The number of horizontal lines were:

>> horz_lines

horz_lines =

2

4.4 DEVELOPING A GENETIC ALGORITHM TOOL

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. The GA tool has been implemented in four steps:

1. Creation of initial population: Choosing random population members within the possible range created the initial population. In our case, the possible range is from Ascii value of A to Ascii value of Z. For simplicity point of view, to apply GA, these values are converted to their binary equivalent before processing and later on converted back to their decimal value.

E.g.: When this step was used, considering the parameters as:

Population size = 10;

Lower limit = 65 (Ascii value of ‘A’);

Upper limit = 90 (Ascii value of ‘Z’);

The answer was:

init_pop =

89

71

80

77

88

84

76

65

86

81

2. Reproduction: It generates a mating pool by selecting good fitness strings from the population. Although there are numerous reproduction operators, but the essential idea in all of them is the same: strings with fitness's above average are picked from the current population and strings with fitness's below average are removed from the population. This procedure may involve maintaining multiple copies of good strings in order to keep the population size fixed. The reproduction operator acts as a filtering mechanism for the selection of the good strings in a population.

In our case, we have used *tournament selection* because of its simplicity. In a typical tournament selection two strings are randomly chosen from the population, and the fitter of the two is selected for insertion into the mating pool.

E.g.: When it was applied on the above initial population, for the character 'B', the results obtained were:

rep_pop =

89

71

87

77

80

84

71

65

84

81

3. Crossover: After reproduction, the crossover operator is applied to strings of the mating pool. Once again, there are a number of crossover operators, but almost all of

them pick two strings from the mating pool at random and then exchange some portion of the strings. In a single point crossover operation, a crossover site is chosen at random and all bits to the right of the crossover site are exchanged between the two strings as shown below:

$$\begin{matrix} 110|010 \\ 100|110 \end{matrix} \Rightarrow \begin{matrix} 110|110 \\ 100|010 \end{matrix}$$

in such an exchange operation, good sub strings from either parent string can be combined to form a better child string provided the right crossover site is chosen. Since we don't know the best crossover site in advance, a random site is chosen. This is simple to implement but random crossover sites can generate children strings that have a poorer fitness than the parents themselves. This is, however, not a problem, since the GA will automatically eliminate such poor quality strings during the next reproduction cycles.

In our case, we have used two-point crossover operator, two sites along the string are chosen at random and the sub strings included between these sites are exchanged between the parents. Single point crossover preserves the maximum amount of information between generations, but two-point crossover gives better search capability.

Normally, crossover is not performed on the entire population. A crossover probability of pc dictates that p_c x 100 percent strings in the population are used in the crossover operation and that the best (1- p_c) x 100 percent of the population are simply copied to the new population,. This preserves some of the better strings for the next generation.

In our case we have taken p_c to be 0.8**.**

Now, applying crossover operator on the above generation yielded following results:

xoverKids =

89

87

77

87

89

84

71

65

76
87

4. Mutation: In addition to the crossover operator, a mutation operator is also used to enhance the search in a GA. The mutation operator flips a bit in a string with a very small *mutation probability*, p_m. Mutation is necessary to maintain diversity in the population, which would otherwise converge very quickly to very similar strings. For e.g., assume that in a specific bit position every string in the population has a value 1. Further, assume that a 0 is required in that position in order to obtain the optimum. Then neither the reproduction nor the crossover operator described will be able to create a 0 in that position. It is only the application of a mutation operator that introduces a finite probability that 1 flipping into a 0.

In our case we have taken p_m to be 0.2.

Now, applying mutation operator on the above generation yielded following results:

mutKids =

89
87
87
77
80
84
88
65
84
81

4.5 DEVELOPING A FITNESS FUNCTION FOR THE GENETIC ALGORITHM TOOL BASED ON THE FEATURES EXTRACTED

The *fitness function* is the objective one want to minimize using GA. The fitness function used in our case was based on shape number. A random initial population was generated using creation step. Now one by one the shape number of the characters in this population was calculated using steps 1 & 2 of the algorithm. The shape number of the 'data' image was also calculated on the same steps. Then the two shape numbers were matched, and deviation was found. If the deviation came to be zero, the character was recognized. Else, the current population was used to create the children that make up the next generation.

The genetic algorithm creates three types of children for the next generation:

- *Elite children* are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.
- *Crossover children* are created by combining the vectors of a pair of parents.
- *Mutation children* are created by introducing random changes, or mutations, to a single parent.

The schematic diagram shown in Fig. 4.11, illustrates the three types of children:

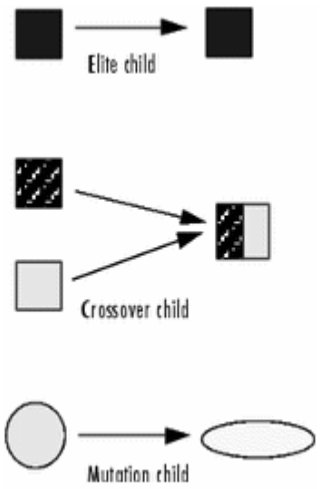
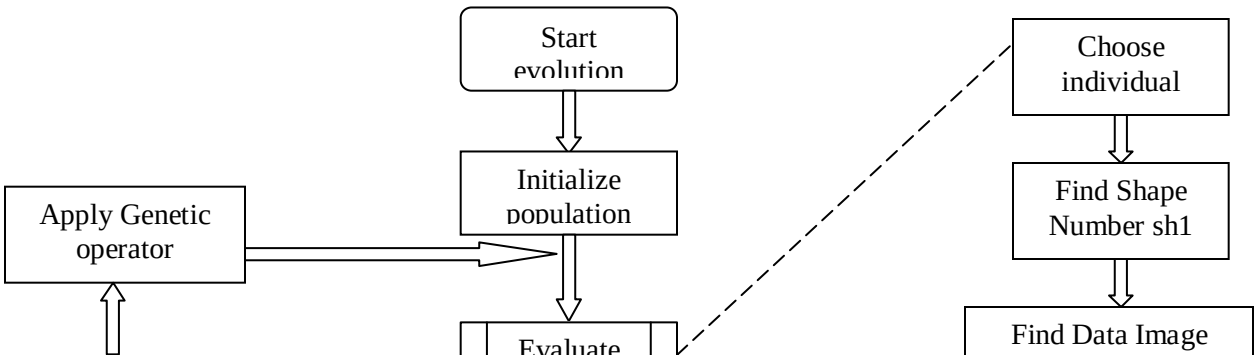


FIGURE 4.11 Three types of children generated by GA.

Now, the same process is repeated for the new population. These steps are repeated recursively until a match is obtained or the generations exceed a predetermined value[13]



Yes

FIGURE 4.12 Flowchart for implementation of Genetic Algorithm

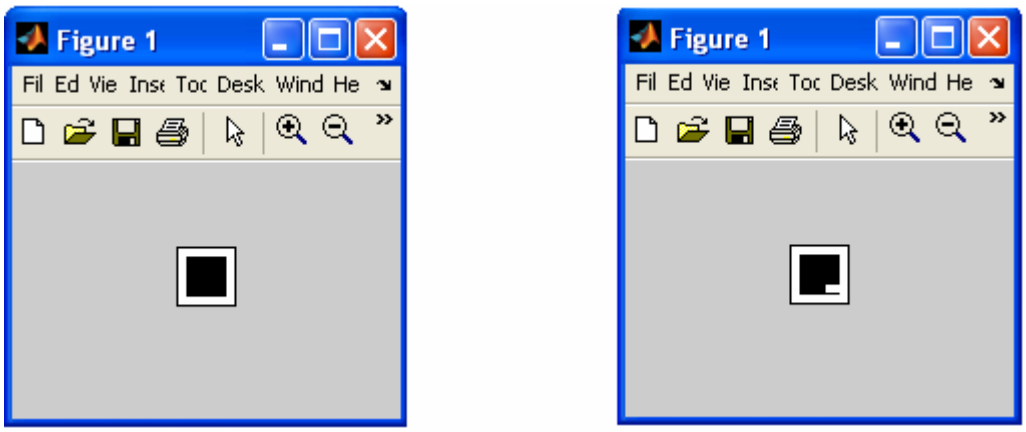
4.6 DETERMINATION OF THE CHARACTERS WITH THE BEST RECOGNITION ACCURACY

This is the last step in the character recognition process. The purpose of including this step was to recognize the character with best accuracy. It was necessary to include tis step, as few of the characters have the same shape and hence the same shape number. These characters cannot be distinguished only by using shape numbers. The other features extracted, in the feature extraction step were used to distinguish these similar characters.

The shape number of ‘O’ and ‘Q’; ‘B’ and ‘D’ were found to be same. So, these characters were distinguished based on the difference in the number of horizontal, vertical or slant lines in their shapes.[15]

The steps for distinction between 'O' and 'Q' are:

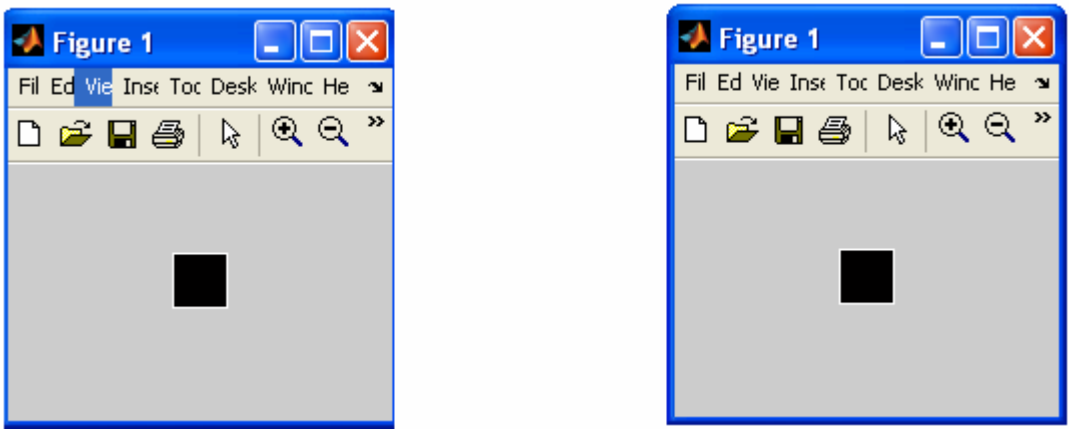
The original images of ‘O’ and ‘Q’ appears as follows:



(a) (b)

FIGURE 4.13 (a) Original Image of 'O'. (b) Original Image of 'Q'

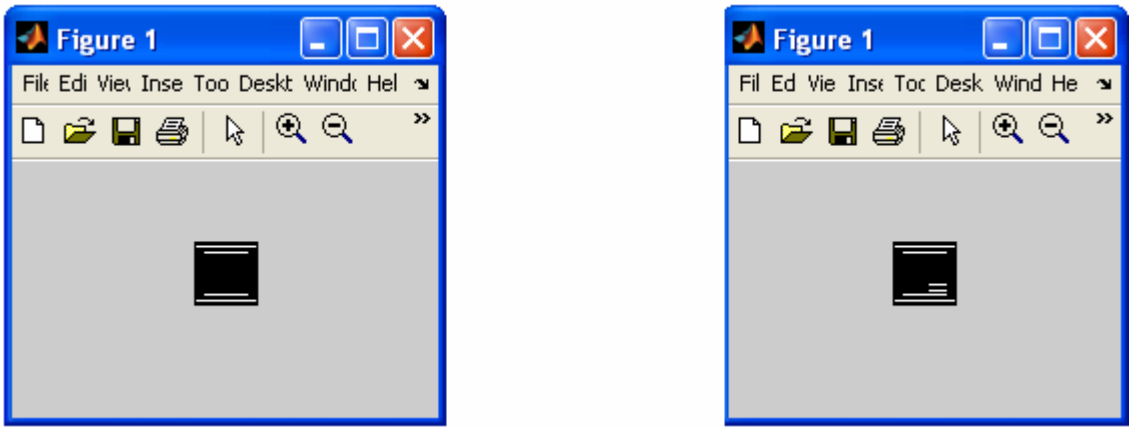
Fig. 4.14 gives their shapes:



(a) (b)

FIGURE 4.14 (a) Shape of 'O'. (b) Shape of 'Q'.

Thus, it can be clearly seen that both are having the same shapes. But if original images are considered, 'Q' is found to have an extra horizontal line as is evident from the following pictures, which are obtained using horizontal mask on the images. This is shown in Fig. 4.15.



(a) (b)

FIGURE 4.15 (a) Horizontal Lines in ‘O’ (b) Horizontal Lines in ‘Q’

When number of horizontal lines was calculated in both of them, results were as:

For ‘O’:

```
>> horz_lines
horz_lines =
    2
```

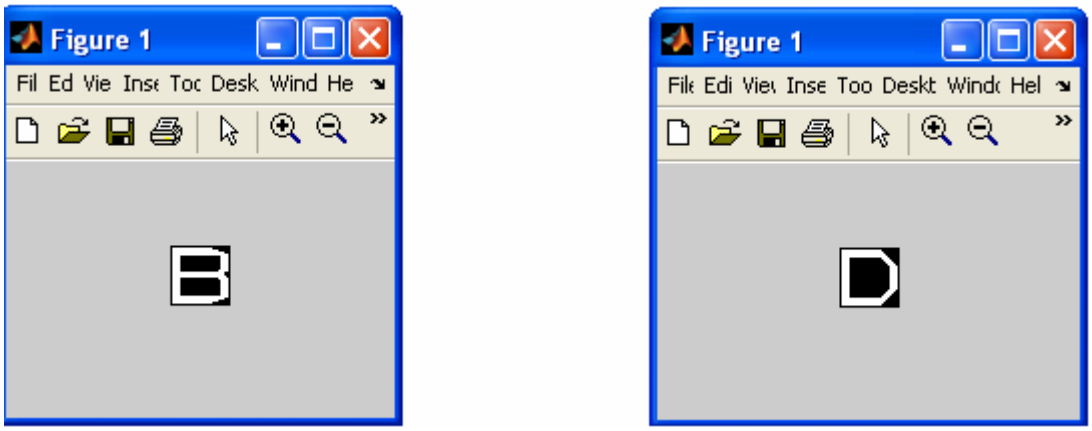
For ‘Q’:

```
>> horz_lines
horz_lines =
    3
```

Thus, the two characters can be distinguished on the basis of their number of horizontal lines.

The steps for distinction between 'B' and 'D' are:

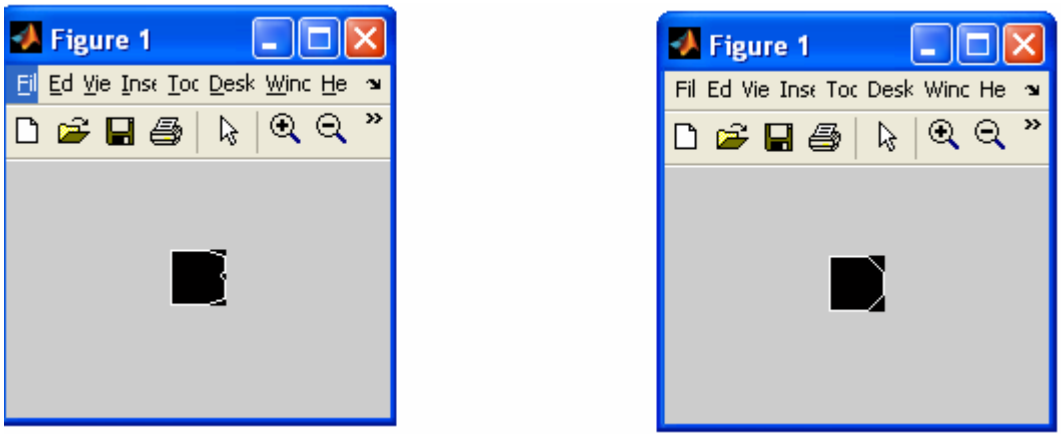
The original images of ‘B’ and ‘D’ appear as shown if Fig. 4.16.



(a) (b)

FIGURE 4.16 (a) Original Image of ‘B’ (b) Original Image of ‘D’

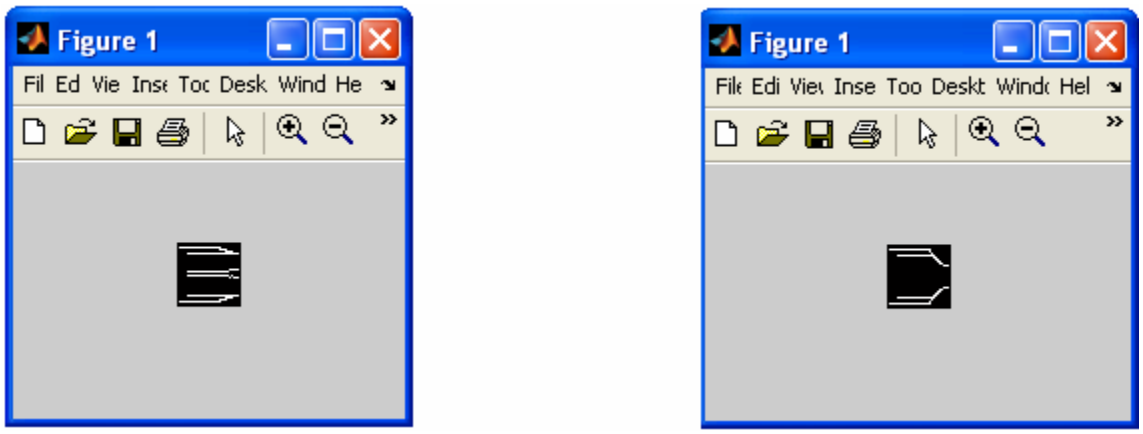
Fig. 4.17 shows their shapes.



(a) (b)

FIGURE 4.17 (a) Shape of ‘B’. (b) Shape of ‘D’.

Thus, it can be clearly seen that both are having almost the same shapes, and when the shape number will be calculated after sampling, it will come exactly same. But if original images are considered, ‘Q’ is found to have an extra horizontal line as is evident from the following pictures, which are obtained using horizontal mask on the images. This is shown in Fig. 4.18.



(a) (b)

FIGURE 4.18 (a) Horizontal Lines in ‘B’ (b) Horizontal Lines in ‘D’

When number of horizontal lines was calculated in both of them, results were as:

For ‘B’:

>> horz_lines

horz_lines =

3

For ‘D’:

>> horz_lines

horz_lines =

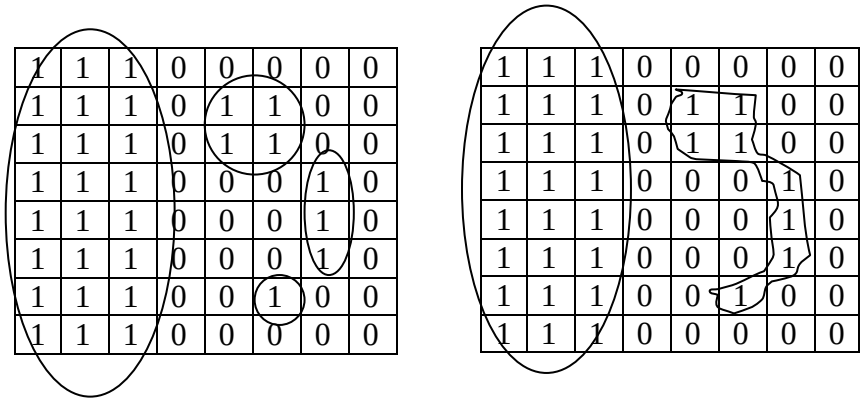
2

Thus, the two characters can be distinguished on the basis of their number of horizontal lines.

4.7 MULTIPLE CHARACTER RECOGNITION

This was the additional feature included. To recognize the multiple characters in a single line the characters were separated and then the same process was implemented on each separated character. To separate the characters the connectivity principle was used.

The Figs. 4.19(a) & (b) show, 4-connected and 8-connected components respectively. Here since background pixels are all '0', we can separate the components from the background. Also if somehow different numbers can be given to different components, as shown in Fig 4.19(c) & (d), the job of separating different characters is almost over.



1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

(a)

1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

(b)

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	4	0
1	1	1	0	0	0	4	0
1	1	1	0	0	0	4	0
1	1	1	0	0	3	0	0
1	1	1	0	0	0	0	0

(c)

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	2	0	0
1	1	1	0	0	0	0	0

(d)

FIGURE 4.19 Connected components (a) four 4-connected components. (b) Two 8-connected components. (c) Label matrix obtained using 4-connectivity. (d) Label matrix obtained using 8-connectivity.

By using this logic, the characters separated on the same line are shown below:
The input image in which multiple characters were clubbed together is shown in Fig. 4.20

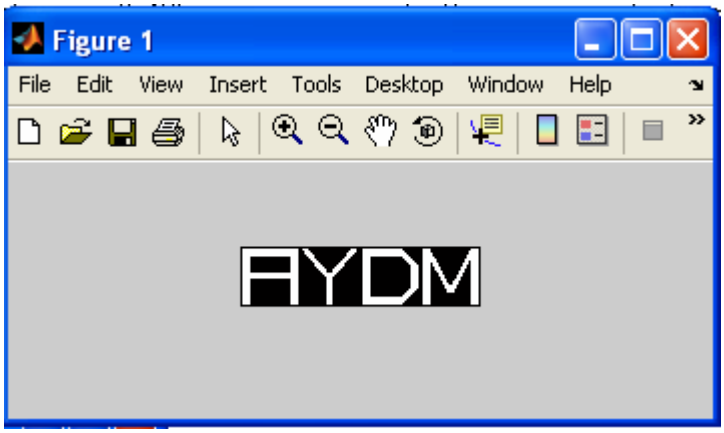


FIGURE 4.20 Input image with multiple characters.

The images obtained for individual characters, after separating them are shown in Fig. 4.21.

(a)

(b)

(c)

(d)

FIGURE 4.20 Images obtained for separated characters.

Now, since the characters have been separated, character recognition can be performed independently on each one of them.

4.8 ROTATIONAL INAVARIANCY

This was also the additional feature included. Due to, rotational inavariancy even if the character was rotated by 90º, 180º or 270º can be recognized. This relives from the burden of placing the image in the correct direction while scanning.

To make the shape number rotation invariant, one of the descriptor of boundary, length was used. The length of a boundary is one of its simplest descriptors. The length of a 4-connected boundary is simply the number of pixels in the boundary, minus 1. If the boundary is 8- connected, we count vertical and horizontal transitions as 1, and diagonal transitions as √2.

The length can be used to find diameter of the boundary. The diameter of a boundary is defined as the Euclidean distance between the two farthest points on the boundary. These points are not always unique, as in a circle or a square, but generally the assumption is that if the diameter is to be a useful descriptor, it is best applied to boundaries with a single pair of farthest point. (When more than 1 pair of farthest points exists, they should be near each other and be dominant factors in determining boundary shape.) The Line segment connecting these points is called the major axis of the boundary. The minor axis of a boundary is defined as the line perpendicular to the major axis and of such length that a box passing through the outer 4 points of intersection of the boundary with the two axes completely encloses the boundary. This box is called the *basic rectangle* and the ratio of the major to the minor axis is called the *eccentricity* of the boundary.

The shape number of a boundary, generally based on 4- directional Freeman chain codes, is defined as the first difference of smallest magnitude. The order of a shape number is defined as the number of digits in its representation. 4- directional Freeman chain codes can be made insensitive to starting point by using the integer of minimum magnitude and made insensitive to rotations that are multiples of 90° by using the first difference of the

code. Thus shape numbers are insensitive to the starting point and to rotations that are multiples of 90°.

If the character image is shown by the Fig. 4.21.

FIGURE 4.20 Character image

Then the rotated image of the boundary is shown by Fig. 4.21.

FIGURE 4.21 Rotated image

But here since the character is in square, thus there will be many diameters. But this discrepancy can be resolved easily, as the shape number will be the same, except that the rotated values of shape numbers will be there.

Rotational inavariancy also introduces confusion of some more characters, if recognized only on the basis of the shape number. These characters are

‘N’ and ‘Z’

‘C’ and ‘U’

‘M’ and ‘W’

These characters were distinguished based on the difference in the number of horizontal,

vertical or slant lines in their shapes.

The steps for distinction of 'N' and 'Z' when rotated are shown as below:

The 'N' in its normal form looks like Fig. 4.22.

FIGURE 4.22 'N' in its normal form**.**

'Z' when rotated at 270° looks like follows Fig. 4.23.

FIGURE 4.23 'Z' when rotated at 270°

Since both pictures appear exactly same their shape number is also same. so, one cannot distinguish based on shape numbers.

But if their corresponding matrices are analyzed, they have values of some pixels different, at the points where slant lines begin. These matrices cannot be shown here because of their large sizes, (they are 30 X 30 matrices). But if number of lines were calculated, then because of these changed pixel values some basic difference was found in their number of lines. This difference is shown below:

In 'N' the lines are:

```
>> horz_lines
horz_lines =
    4
>> vert_lines
vert_lines =
```

2

and in 'Z' the lines come out to be:

```
>> horz_lines
horz_lines =
    5
>> vert_lines
vert_lines =
    2
```

Thus, as clear from above results, the two characters can be distinguished based on horizontal number of lines.

The steps for differentiation of 'M' and 'W' when rotated is shown as below:

The 'M' in its normal form looks like Fig. 4.24

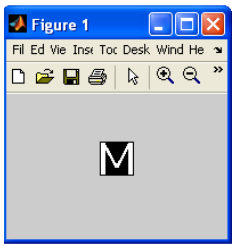


FIGURE 4.24 'M' in its normal form**.**

The 'W' when rotated by an angle of 180° looks as shown in Fig. 4.25.

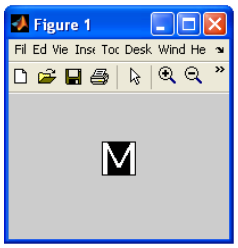


FIGURE 4.25 'W' when rotated at 180°

Again, since both pictures appear exactly same their shape number is also same. So, these characters cannot be distinguished based on their shape numbers.

If the two matrices were compared at pixel level, it was found that their matrices were not exactly equal; rather there was some difference at the starting points of the slant lines. Thus when numbers of lines were calculated, following results were obtained:

For ‘M’:

```
>> pos_slant_lines
pos_slant_lines =
    0
>> neg_slant_lines
neg_slant_lines =
    5
```

For rotated ‘W’:

```
>> pos_slant_lines
pos_slant_lines =
    1
>> neg_slant_lines
neg_slant_lines =
    3
```

Thus, as clear form above, 'M' and 'W' can be differentiated based on the number of positive or negative inclined lines, in their shape.

The 'C' and 'U' when rotated cannot be distinguished, as not only their shape but matrices also become exactly equal after rotation.

CHAPTER 5

EXPERIMENTAL RESULTS AND SYSTEM OVERVIEW

WHEN THE INPUT IMAGE CONTAINED SINGLE CHARACTER:

Input/Data Image:

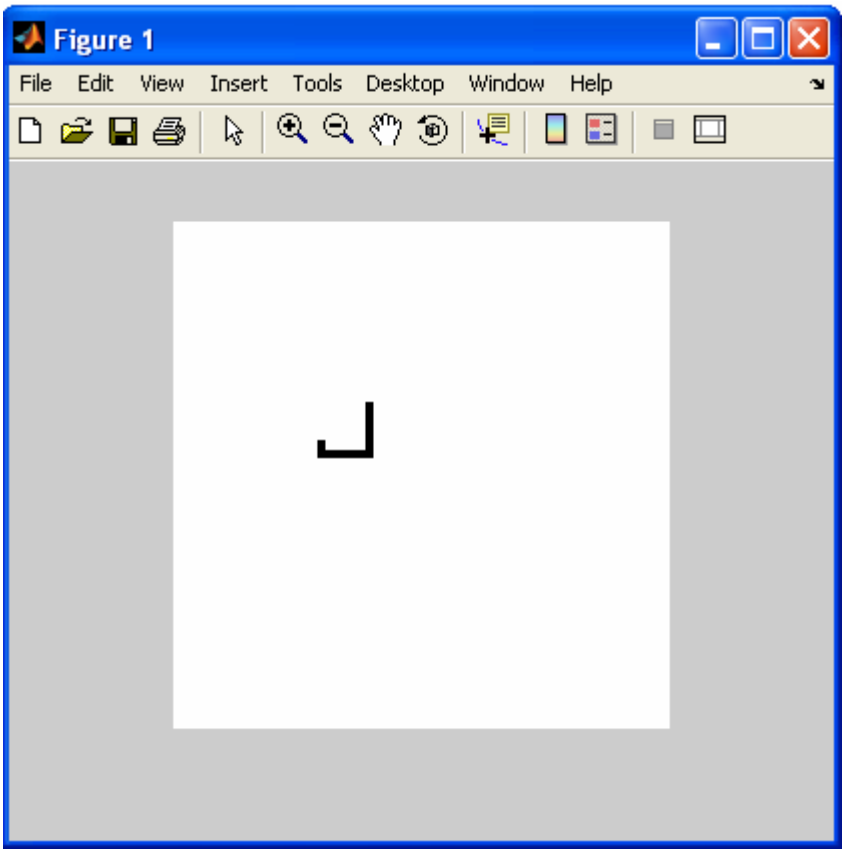


FIGURE 5.1 Input Image

Processed Input Image:

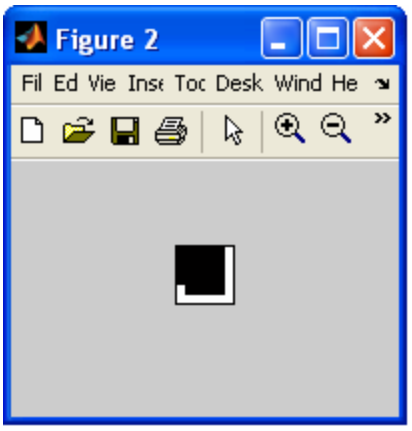


FIGURE 5.2 Processed Image

Image after Extraction of Boundary:

FIGURE 5.3 Boundary of the input image

Image after Rotation of Boundary along the Diameter is:

FIGURE 5.4 Rotated Image

The shape number of the given boundary is:

0 2 6 6 6 2 6 6 2 6 6 4 2 2

The total number of generations undergone is:

11

ans =

The 1th character is J

>>

WHEN THE INPUT IMAGE CONTAINED SINGLE ROTATED CHARACTER:

Input/Data Image:

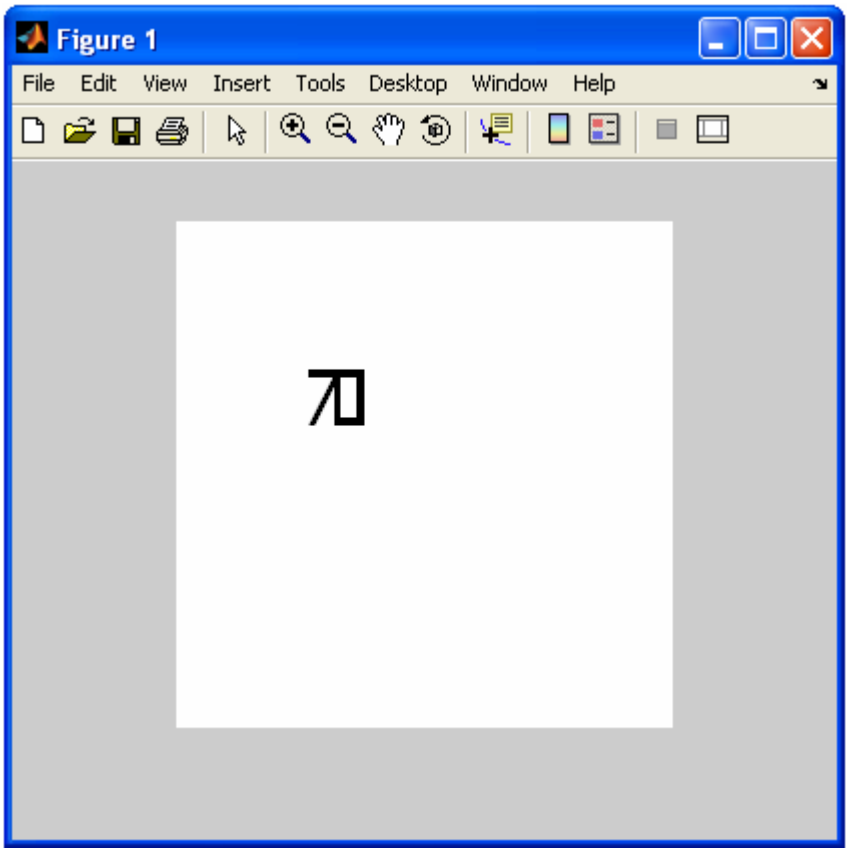


FIGURE 5.5 Input Image

Processed Input Image:

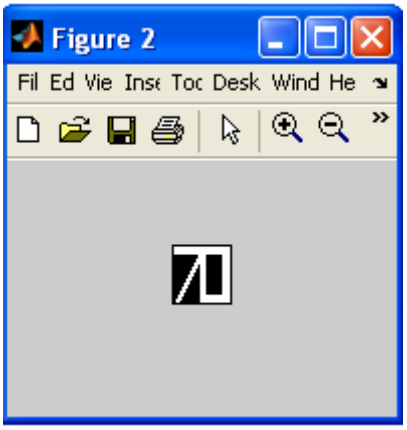


FIGURE 5.6 Processed Image

Image after Extraction of Boundary:

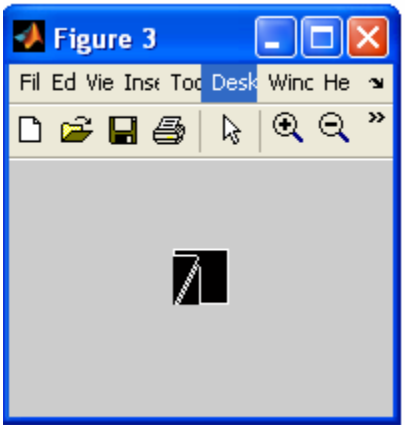


FIGURE 5.7 Input Image

Image after Rotation of Boundary along the Diameter is:

FIGURE 5.8 Boundary of the input image

The shape number of the given boundary is:

2 6 6 2 4 2 6 6 4 0 4 0 2 4

The total number of generations undergone is:

3

ans =

The 1 th character is R

>>

WHEN THE INPUT IMAGE CONTAINED MULTIPLE CHARACTERS:

Input/Data Image:

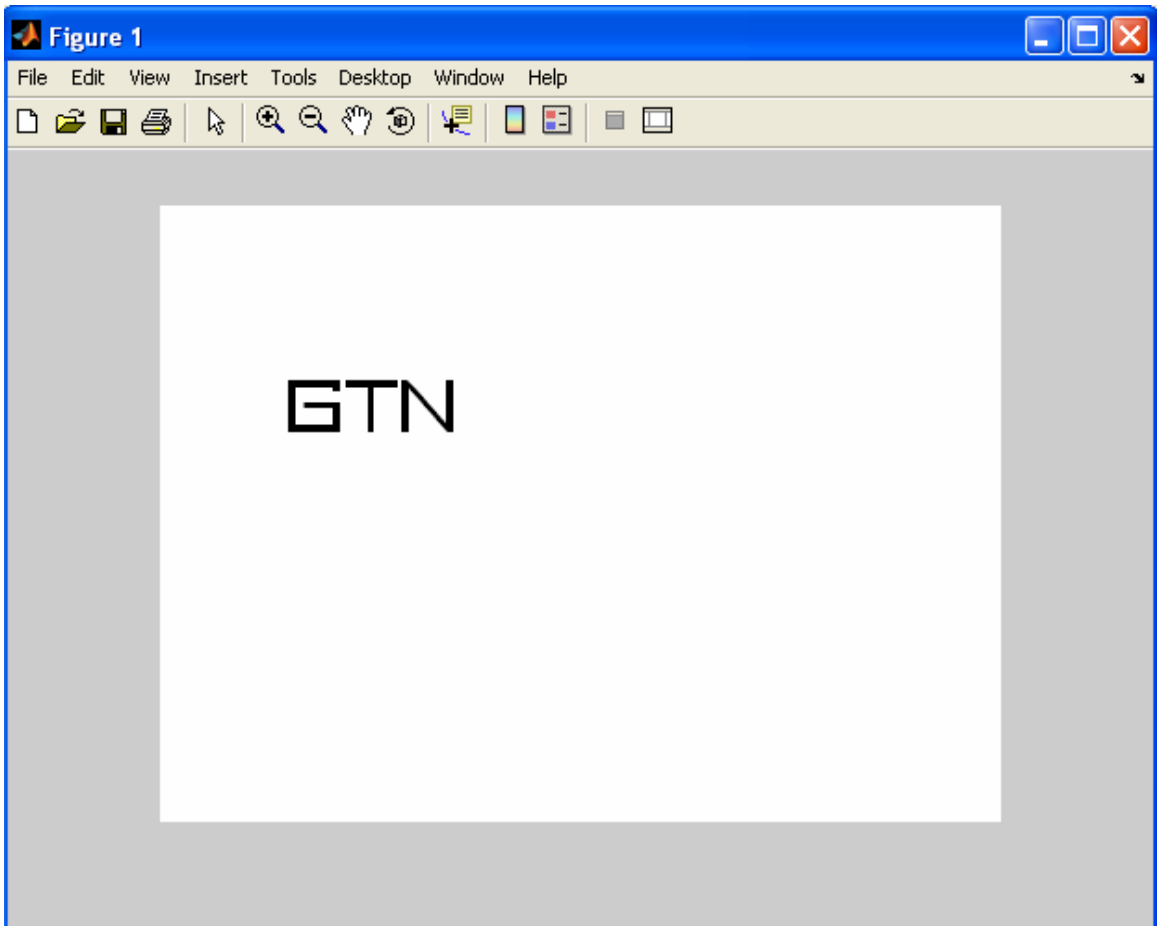


FIGURE 5.9 Input Image

Processed Input Image:

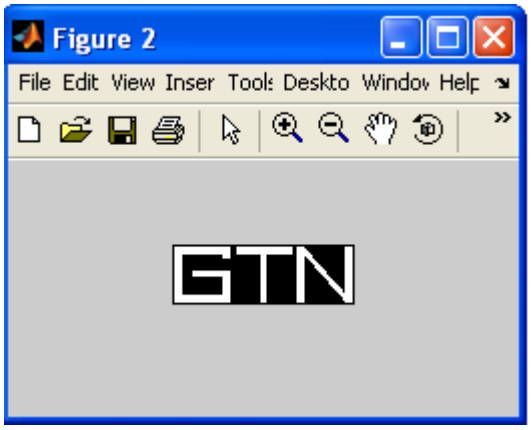


FIGURE 5.10 Processed Image

Boundary Images of the Separated Characters are:

FIGURE 5.11 Separated Characters

Their Respective Images after Rotation of Boundary along the Diameter are:

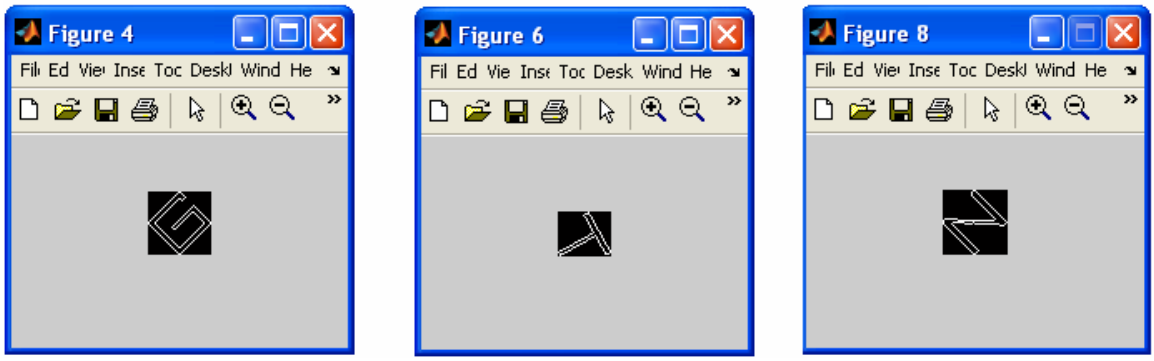


FIGURE 5.12 Boundaries of the rotated Images

The respective characters that were recognized are given as:

The shape number of the given boundary is:

Columns 1 through 16

0 4 2 6 2 4 2 6 6 2 6 4 2 6 3 1

Columns 17 through 21

2 2 2 4 6

The total number of generations undergone is:

0

ans =

The 1 th character is G

The shape number of the given boundary is:

2 5 7 4 2 2 6 6 6

The total number of generations undergone is:

17

ans =

The 2 th character is T

The shape number of the given boundary is:

0 4 6 6 6 2 4 0 0 3 7 4 2 6 6

The total number of generations undergone is:

1

ans =

The 3 th character is N

>>

CHAPTER 6

CONCLUSION AND SCOPE FOR FUTURE WORK

In this study, a tool has been developed using genetic based fitness module to solve the problem of Character Recognition. This tool has been previously applied to a variety of problems, but character recognition seems to be complex. This may be due to the high amount of information needed to classify a character correctly and because of the high degree of similarity among character shapes. Nonetheless, an encouraging solution to the problem has been found, by using a feature extraction module, which absorbs much of the misleading similarity, and a genetic based fitness module, which produces descriptions of character shapes, discriminating between features essential for distinguishing among different character classes.

Genetic techniques have the facility of discovering novel solutions to the problems using a subtle mixture of user defined elementary operations, fitness driven selection, pure luck, and brute force. The technique was modeled using MATLAB simulator tool.

The experimental result of this method for recognizing the character strings has shown promising performance. The character strings have been recognized with 100% accuracy, incorporating rotational invariability.

The *future work* would encompass broadening the character recognition tool thereby including font size invariability and recognition of handwritten characters.

BIBILIOGRAPHY

1. Tinku Acharya and Ajoy K. Ray, “Image Processing Priciples and Applications”, Wiley Interscience.
2. Rafal C.Gonzalez and Richard E.Woods, “Digital Image Processing” Addision Wesley publishing company, 1993, ISBN –201-50803-B.
3. Anil K.Jain, fundamentals of Digital Image Procession, Pearson Education, 1989 ISBN 81-297-0083-2.
4. Azriel Rosenfeld and Avinash C.Kak, “Digital Picture Processing”, Academic Press, INC, 1982, San Diego, California 92101.
5. Earl Gose & Richard Johnsonbaugh & Steve Jost, “Pattern Recognition and Image Analysis”, Prentice Hall India.
6. David E. Goldberg, “Genetic Algorithm in search Optimization and Machine Learning”, Pearson Education.
7. Pattern Recognition from Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Pattern.
8. W. E. Dickinson, “A Character Recognition Study”, IBM journal, July 1960.
9. Maria Petrou & Josef Kittler, “Optimal Edge Detectors For Ramp Edges”, IEEE trans on Pattern Analysis & Machine Intelligence, vol-13, no-5, May 1991.
10. Anil K.Jain, Robert P.W. Duln and Jiancharng Mao “Statisfical Pattern Recognition : A Review” IEEE trans Pattern Analysis and Machine Intelligence, vol-22, no-1, January 2000, pp 4-34.
11. K. G. Khoo and P. N. Suganthan “Structural Pattern Recognition Using Genetic Algorithms with Specialized Operators”, IEEE tran Systems, Man, and Cybernetics, Vol 33 No.1 , February 2003.
12. Marcin Borkowski, “Application of Genetic Algorithm to Pattern Extraction”, International conference on Intelligent Systems Design and Applications (ISDA’05), IEEE 2005.

13. Yuan-Kai Wang and Kuo-Chin Fan, “Applying Genetic Algorithms on Pattern Recognition: an analysis and Survey”, Proceeding of ICPR’ 96, IEEE 1996.

14. Louis A. Tamburino and Michael A.Zmudu, “Generating Pattern Recognition Systems using Evolutionary learning”, IEEE 1995.

15. Naoki Saito and M.A. Cunningham, “Generalized E-Filter and Its application to Edge Detection”, IEEE trans Pattern Analysis and Machine Intelligence, vol-12, no-8, August 1990.

16. Richard N.Czerwinski, Douglas L.Jones & William D.O’Brien “Line and Boundary Detection in Speckle Images”, IEEE trans Image processing, Vol-7 No. 12, December 1998.

17. Alexandre Lemieux, Christian Gagne and Marc Parizeau, “Genetical Engineering of Handwriting Representations”, International workshop on Frontiers in handwriting Recognition, IEEE 2002.

18. Bruno Botempi and Angelo Marcelli, “Machine learning and genetic algorithms: an application to character recognition”, IEEE 1994.

19. L.S Oliveria N.Benahmed, R.Sabouria , F. Bortolozzi, C.Y. Suen, “Feature subset Selection Using Genetic Algorithms for Handwriting Digital Recognition”.

20. Bruno Botempi and Angelo Marcelli, “A Genetic Learning System for On-line Character Recognition”, IEEE 1994.