A WRAPPER BASED APPROACH FOR PERSONAL IDENTIFICATION THROUGH KEYSTROKE DYNAMICS USING SOFT COMPUTING TECHNIQUES

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION
3.2 PROBLEM SPECIFICATION
3.3 METHODOLOGY OVERVIEW
3.4 FEATURE EXTRACTION IN KEYSTROKE DYNAMICS
3.5 PROPOSED FEATURE–VIRTUAL KEY FORCE–PHASE I
3.6 CONCLUSION OF PHASE I
3.7 NORMALIZATION–PHASE II
3.8 CONCLUSION OF PHASE II
3.9 FEATURE SUBSET SELECTION
3.10 PROPOSED WRAPPER BASED FEATURE SUBSET SELECTION–PHASE III
3.11 CONCLUSION OF PHASE III
3.12 CLASSIFICATION
3.13 EXTREME LEARNING MACHINE (ELM)
3.14 MODIFIED LEVENBERG-MARQUARDT ALGORITHM
3.15 ANALYTIC NETWORK PROCESS (ANP)
3.16 PROPOSED WRAPPER BASED CLASSIFICATION WITH ELM-ANP APPROACH–PHASE IV
3.17 CONCLUSION OF PHASE IV
3.18 PROPOSED WRAPPER BASED ELM-ANP FOR FEATURE SUBSET SELECTION
3.19 CONCLUSION OF ELM-ANP WRAPPER BASED FEATURE SUBSET SELECTION
CHAPTER 3

METHODOLOGY

3.1. INTRODUCTION

A Keystroke represents a key action of the user on the keyboard and provides detailed timing information about exactly when each key is pressed and released as a person is typing at a computer keyboard.

This chapter provides the detailed overview of the research methodology and the techniques used in the present research work.

3.2. PROBLEM SPECIFICATION

Over the last two decades, different features, feature extraction, feature subset selection and classification methods have been introduced and studied by researchers in order to improve the recognition capabilities of keystroke based authentication mechanism. For feature extraction, Virtual Key Force (VKF), a new feature is suggested for better performance. The proposed methodology uses wrapper approach based on Extreme Learning Machine with Genetic Algorithm, Ant Colony Optimization and Particle Swarm Optimization for feature subset selection. Further, selected features are classified by Extreme Learning Machine with Analytic Network Process which improves the classification accuracy of the system tremendously and reduces the training time and testing time required. The ELM-ANP method which is used for classification is tested with wrapper based feature subset selection which also reduced the number of features selected compared with the other methods.
3.3. METHODOLOGY OVERVIEW

The proposed method is attempted to provide better classification accuracy and to reduce training and testing times. The overview of methodology is shown in figure 3.1.

![Figure 3.1 Methodology Overview](image)

The entire methodology is divided into four phases namely, feature extraction phase, normalization phase, feature subset selection phase and classification phase after obtaining the raw keystroke data (press time and release time). The significant contributions are done in feature extraction, feature subset selection and classification phases.

3.3.1. Feature Extraction

The feature extraction characterizes the attributes common to all patterns belonging to a class. The key function of feature extraction in keystroke dynamics is to extract the fundamental features from the timestamp collected from raw keystroke data for creation of template. For the benchmark datasets [87, 88] and real time dataset used in the study, the features Duration or Dwell time, Flight time or Latencies, Digraph and Trigraph are extracted. In addition to all the timing features, a new feature called Virtual Key Force has been introduced.
3.3.2. Normalization

The typing pattern of the user varies from time to time even for the same user. There will also be a significant difference in the keystroke patterns exhibited from person to person, even when asked to type the same words [20]. Therefore data normalization is done between 0 and 1 to preprocess the data. Z-Score, Min-Max, Zero Mean and Standard Deviation, Median - MAD, and Tanh normalizations are done and Z-score normalization is found to be effective compared to other normalization methods.

3.3.3. Feature Subset Selection

Feature subset selection identifies the most selective features. It reduces the dimensionality of features which improves the accuracy and decreases the computation time. Feature subset selection is fundamentally an optimization problem, which concerns searching the space of possible features to be familiar with one that is optimum or near-optimal in accordance with some performance measure [65]. The purpose is to obtain any subset that reduces or to improve a particular measure. After preprocessing with Z-Score normalization, feature subset selection is done for reducing the features for further processing. Some of the optimization techniques that are suggested to select the subset of features from extracted features are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA). These techniques are wrapped with Extreme Learning Machine (ELM) which can routinely select an appropriate subset of features and the rest will not be considered, thus concluding in a more comprehensive model.

3.3.4. Classification

The purpose of classification is to find the best class that is closest to the classified pattern. The selected features are given as input to classifiers such as
Back Propagation Neural Network (BPN), Extreme Learning Machine, and Extreme Learning Machine with Analytic Network Process (ELM-ANP). The wrapper based classification approach using PSO-ELM-ANP outperforms the other classifiers used in this study with the average accuracy of 99.92% and it reduces the training and testing time taken compared to other methods. The contributions of the proposed work are shown in Figure 3.2.
Sections 3.1 to 3.3 explained the methodology proposed in the research work. The problem specification and the contributions of the proposed work are elaborated in this section. The next section discusses about the new feature called Virtual Key Force (VKF) which is the contribution done in feature extraction phase.

3.4. FEATURE EXTRACTION IN KEystroke DYNAMICS

Feature extraction is the first step to be done after obtaining the raw keystroke data. It is the process of extracting vital information in any given problem. All the common patterns belonging to a class are characterized by feature extraction. A complete set of biased features for each pattern class can be found using feature extraction.

Extraction of fundamental features from the timestamp collected from raw keystroke data for creation of template is the key function of feature extraction in keystroke dynamics. Keystroke data can be obtained by measuring the pressing and releasing time of keys. There are many features that can be measured from the keystrokes. Some of the significant features that can be measured from keystroke are as follows:

- Dwell Time (DT),
- Flight Time (FT),
- Pressure of keystroke,
- Difficulties of typing text,
- Frequency of word errors and
- Typing rate.

All the features are not useful and are not widely used and only a few features are frequently used [61]. For measuring pressure and force of keystroke, special type of pressure or force sensitive keyboard is required. Difficulties of
typing text, frequency of word errors, typing rate are useful only for long text. Since user will be providing only password these features are not suitable. Therefore the timing feature such as Duration or Dwell time, Latency or Flight time, Digraph, Trigraph are commonly measured from keystroke. The following Figure 3.3 shows the timing features that can be measured from the keystroke.

Figure 3.3 Timing Features from the Keystrokes

The features that can be extracted from timing vector of a keystroke are briefed below:

**Dwell Time or Duration (DT)**

Dwell time refers to how long a key is pressed until it is released. It is the difference between time of pressing a key and releasing the same key and can be calculated as

\[ DT = R1 - P1 \]  

(3.1)

where R1 indicates a key release and P1 indicates a key press.
Flight Time (FT) or latencies

Flight time is the interval time between key press and release of different keys. It is categorized as follows:

a) Release-Press (RP)

RP is the interval between a key release and next key press time and it is calculated from the following equation:

\[ RP = P2 - R1 \]  

(3.2)

where \( P2 = \) press time of next key and \( R1 = \) Release time of first Key.

b) Press-Press (PP)

PP which is also called the Digraph which is the interval between a key Press and next key Press and it is calculated from the following equation:

\[ PP = P2 - P1 \]  

(3.3)

where \( P2 = \) press time of next key and \( P1 = \) press time of first Key.

c) Release-Release (RR)

RR is calculated from the Interval between two successive key Releases and always has the positive value. It is given as follows:

\[ RR = R2 - R1 \]  

(3.4)

where \( R2 = \) Release time of second key and \( R1 = \) Release time of first key.

d) Press - Release (PR)

PR is the Elapsed time between Pressing of first key and releasing of next key. It is calculated from the following equation:
\[ PR = R2 - P1 \] (3.5)

where \( R2 \) = Releasing of second key and \( P1 \) = Pressing of first key

**Trigraph (TRI)**

Trigraph is the Elapsed time between the first key press and the third key press. It is given as follows:

\[ TRI = P3 - P1 \] (3.6)

where \( P3 \) is the pressing of third key and \( P1 \) is the pressing of first key.

### 3.5. PROPOSED FEATURE-VIRTUAL KEY FORCE – PHASE I

Keystroke Dynamics is one of the famous and inexpensive behavioral biometric technologies, which will try to identify the authenticity of a user when the user is working with the keyboard. There are many features that can be acquired using keystroke of the user. Force of key typed is one of the features which can be obtained using a special force sensitive keyboard [50] which is expensive and also may not be available commonly. In addition to the entire timing features such as duration, latency, digraph and trigraphs, a new proposed Feature called Virtual Key Force (VKF) has been introduced. The virtual key force is measured without using any special key board which also improves the accuracy when the feature is used for classification.

The Virtual Key Force is calculated based on the typing speed and the behavior of the user at the keyboard. It measures the time taken by the user between releasing one key and pressing another key. It is based on the fact that each user has different typing speed and each user takes his/her own time to release and press another key. The usage of keys and the typing speed and force are different for different users. Moreover, the time interval taken since the release of one key and press of another key is also different. Consider a user typing a
word which consists of ten letters. Hence there exist nine time intervals between the release of one key and press of another key. The average typing speed of the user can be calculated based on these time intervals. Virtual key force can be determined from the Complexity Label (CL) and Key Complexity (KC). The complexity label can be calculated as follows:

- According to the complexity of usage of the keys, key complexity can be determined. It is based on the key position and distance.
- It means that the middle row keys (i.e., the keys from A to L) on the keyboard which are easy to handle by all the users is taken as 0. The key complexity of remaining keys is taken as 1. For typing the adjacent keys, CL is 0.

In the figure 3.4, for the keys T, H, E the complexity label is assigned as CL = (0,1). i.e. the distance from T and H is nearer(0) and the distance between H and E is longer(1). In the following figure 3.4 key positions in a keyboard and the timing intervals between keys T, H, E are shown.

![Figure 3.4 Timing Intervals between Keys](image)

The Complexity Label (CL) calculation is fully user dependent and it is given in the following table 3.1 for all the passwords used in the thesis. The Complexity Label also depends on the hand used by the user. For example, the CL of the password ‘drizzle’ is determined as follows:
The CL of first character ‘d’ is assigned as 0. For typing the character ‘r’ the left hand finger has to move from middle row to upper row. So CL is 1. The character ‘I’ is in upper row. Therefore CL is 1. For typing ‘z’ the left hand finger has to move from ‘r’ to ‘z’. So CL is 1. For typing the same key again CL is 0. For typing ‘l’ which is in middle row, CL is 0. For typing ‘e’, the left hand finger has to move from ‘z’ to ‘e’. So CL is assigned as 1.

Table 3.1 Complexity Label Calculation

<table>
<thead>
<tr>
<th>Password</th>
<th>Complexity Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>drizzle</td>
<td>0111001</td>
</tr>
<tr>
<td>jeffrey allen</td>
<td>0100101100011</td>
</tr>
<tr>
<td>.tie5Roan1</td>
<td>011111111011</td>
</tr>
<tr>
<td>pass tie.R</td>
<td>0000111111111</td>
</tr>
<tr>
<td>nopassword</td>
<td>0110001110</td>
</tr>
</tbody>
</table>

After calculating the complexity label, the average time is calculated. It is the total time taken divided by the time taken to press a key. If the actual time is less than the average time, then the key complexity is assumed as 0 (i.e Shorter Interval between Keys) otherwise key complexity is assumed as 1 (i.e longer Interval between keys).

Based on the key complexity and the complexity label, the following algorithm which is shown in table 3.2 has been formulated:

Table 3.2 Virtual Key Force Algorithm

```plaintext
if (key distance is nearer || longer && time interval is below average)  
   VKF=3 (low)
else if (key distance is nearer && time interval is above the average)  
   VKF=2(medium)
else if (keys are longer and the average time interval is above the average)  
   VKF=1 (high)
end
```
The three levels of force namely High, Medium and Low are identified. If the key distance is nearer or longer and time interval taken to hit the key is below average then the key is assumed to be hit with low force. If the key distance is nearer and time interval taken to hit the key is above average then the key is assumed to be hit with medium force. If the key distance is longer and time interval taken to hit the key is above average then the key is assumed to be hit with high force. The Virtual Key Force (1 or 2 or 3) for each character is stored as feature for the passwords.

The table 3.3 shows the extracted features from all the passwords in the datasets used in the study.

**Table 3.3 Features Extracted from the Passwords**

<table>
<thead>
<tr>
<th>S.no</th>
<th>Passwords</th>
<th>No. of characters</th>
<th>DT</th>
<th>RP</th>
<th>PP</th>
<th>RR</th>
<th>PR</th>
<th>TRI</th>
<th>VKF</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>drizzle</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>jeffrey allen</td>
<td>13 (including space)</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>13</td>
<td>85</td>
</tr>
<tr>
<td>3</td>
<td>.tie5Roan1</td>
<td>11 (including shift key to press R)</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>71</td>
</tr>
<tr>
<td>4</td>
<td>nopassword</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>pass_tie.R</td>
<td>12 (including shift key to press _ and R)</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>12</td>
<td>78</td>
</tr>
</tbody>
</table>
3.6. CONCLUSION OF PHASE I

A new feature called Virtual Key Force has been proposed which improves the accuracy and also reduces the training and testing time. The Virtual Key Force is an additional proposed feature apart from the commonly available features. When this proposed feature is used with wrapper based approaches, the accuracy is improved. This proves the efficiency of the proposed new feature which is observed during the later part of the study.

Next to feature extraction process, normalization is done. The next section discusses about the normalization techniques applied in the proposed work.

3.7. NORMALIZATION – PHASE II

After feature extraction, preprocessing has to be done. This section focuses on preprocessing using normalization techniques. The time and speed of typing varies for same user even for the same text. The figure 3.5 shows the variations in typing the same text by the same user for 10 times. In order to bring the data in a particular range (0 to 1), data normalization is carried out.

![Figure 3.5 Typing Variations](image-url)
The normalization techniques such as Z-Score, Min-Max Normalization, Zero Mean and Standard Deviation, Median - MAD, and Tanh Normalization which are shown in the figure 3.6 are applied to the extracted features. Z-Score Normalization is found to be more effective than other methods. The unnecessary features contained in extracted feature set are normalized using Z-Score method. Feature subset selection is done with the normalized results as the next step.

![Diagram of Normalization Techniques](image)

**Figure 3.6 Normalization Techniques**

3.7.1. **Z-Score Normalization**

The most commonly used score normalization technique is the z-score normalization which is calculated using the arithmetic mean and standard deviation of the given data. The formula is given as follows:

$$ Z = \frac{x - \mu}{\sigma} \quad (3.7) $$

where $\mu$ = mean, $x$ = matching score, $\sigma$ = standard deviation.

3.7.2. **Min-Max Normalization**

It is the simplest normalization techniques and can be used where minimum and maximum range of the data is known. The equation of Min-Max normalization is given as follows:
\[
Z = \frac{x - \text{min}}{\text{max} - \text{min}}
\] (3.8)

where \(x\)=matching score, \(\text{min}\)= minimum data value and \(\text{max}\)= maximum data value.

### 3.7.3. Zero Mean and Standard Deviation Normalization

The Zero Mean and Standard Deviation (mapstd) normalizes the inputs and targets so that they will have zero mean and unity standard deviation. The formula is given as follows:

\[
Z = (x - x_{\text{mean}}) + (z_{\text{std}}/x_{\text{std}}) + z_{\text{mean}}
\] (3.9)

where \(x_{\text{mean}}\) is the mean of input data and \(z_{\text{mean}}\) is set as 0, \(x_{\text{std}}\) standard deviation of input data and \(z_{\text{std}}\) is set as 1.

### 3.7.4. Median – MAD Normalization

The simplest way to quantify variation is Median-Median Absolute Deviation (MAD). In this normalization technique, initially the median of all the values are calculated. From the median value, the distance among each value is calculated. Whether the value is greater or lesser than the median, the distance between the value and the median is positive. The median of the set of differences is calculated and the resultant value is denoted as Median Absolute Deviation (MAD). The formula of Median – MAD normalization is given in the equation 3.10.

\[
Z = \frac{x - \text{median}}{\text{MAD}}
\] (3.10)

where \(\text{MAD} = \text{median}(|x-\text{median}|)\).
3.7.5. Tanh Normalization

The Tanh normalization is highly efficient and is given by the equation 3.11.

\[ Z = \frac{1}{2} \left\{ \tanh \left( 0.01 \left( \frac{x - \mu}{\sigma} \right) \right) + 1 \right\} \quad (3.11) \]

where \( \mu \) and \( \sigma \) are mean and standard deviation respectively.

3.8. CONCLUSION OF PHASE II

Normalization techniques that are applied for the problem taken for study such as Z-Score, Min-Max, Zero Mean and Standard Deviation, Median-MAD and Tanh are discussed. Next to normalization is feature subset selection. There are various feature subset selection techniques available in the literature. The next section discusses the proposed wrapper based feature subset selection approach which yield better results.

3.9. FEATURE SUBSET SELECTION

Feature subset selection identifies the most discriminating features. It also reduces the dimensionality of features which improves the assumption accuracy and decreases the computation time. Feature subset selection is applied to high dimensional data before preceding the classification since the increased dimensionality of features makes testing and training of classification method difficult.

Feature subset selection is fundamentally an optimization problem. It searches the space of possible features to be familiar with one that is optimum or near-optimal in accordance with some performance measures. The purpose of feature subset selection is to obtain any subset that reduces or to improve a particular measure.
Feature selection techniques are divided into two types namely, feature ranking and feature subset selection. Feature ranking ranks the features by a metric and eliminates all features that do not achieve an adequate score. Subset selection searches for the set of possible features to construct the optimal subset. To carry out this search, a starting point, a strategy to traverse the space of subsets, an evaluation function and a stopping criterion are to be specified. Although there are varieties of subset selection techniques developed, usually two methods namely filter and wrapper approaches are considered. The classification of feature selection is shown in the figure 3.7.

![Figure 3.7 Classifications of Feature Selection Techniques](image)

### 3.9.1. Filter Approach

In Filter approach, the feature subset selection is done independently of the learning algorithm. The subset selection procedure in this case can be seen as a preprocessing step. The Filter approach is shown in the figure 3.8.

![Figure 3.8 Filter Approach](image)
3.9.2. Wrapper Approach

The wrapper approach uses the method of classification as objective function to measure the importance of features set. Hence the feature selected depends on the classifier model used.

In the wrapper approach, the feature subset selection is done by induction algorithm. The feature subset selection algorithm conducts a search for a good subset using the induction algorithm as a part of the evaluation function. The purpose of the induction algorithm is to induce from training data a classifier that will be useful in classifying future cases. The accuracy of induced classifiers is estimated by accuracy estimation technique. The classification algorithm itself is used to determine the attribute subset. Since the wrapper approach optimizes the evaluation measure of the classification algorithm while removing features, it mostly leads to greater accuracy than the filter approaches such as $X^2$ Statistic, Information gain, Symmetrical uncertainty, ReliefF and Correlation based feature selection. Wrapper methods generally result in better performance than filter methods because the feature selection process is optimized by the classification algorithm to be used. Hence a wrapper based feature subset selection approach is proposed in this thesis. The wrapper based approach for feature subset selection is shown in figure 3.9. The next section 3.10 discusses the proposed wrapper based feature subset selection approaches.
3.10. PROPOSED WRAPPER BASED FEATURE SUBSET SELECTION–PHASE III

Feature subset selection is applied to high dimensional data before preceding the classification. Feature subset selection is an optimization problem, which searches the set of possible optimum or near-optimal features, since the objective is to acquire any subset that reduces or improves a particular measure.

Optimization techniques like Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Ant Colony Optimization (ACO) are wrapped with Extreme Learning Machine (ELM) to select the subset of features from the extracted features. These proposed wrapper based approaches select an appropriate subset of features and the rest will not be considered, thus resulting in a more comprehensive model.

3.10.1. Extreme Machine Learning (ELM)

Extreme Learning Machine is used as induction algorithm in PSO, GA and ACO for wrapper based feature subset selection. BPN and SVM are the commonly used learning algorithms. BPN and SVM require manual tuning which degrades the performance of the system. ELM on the other hand does not require any manual tuning and avoids the issues such as local minima, improper learning rate.
and overfitting, etc., faces by other learning algorithms. The Extreme Machine Learning algorithm is narrated below:

If there are $N$ samples/features $(x_i, t_i)$ with $x_i \in R^n$, $t_i \in R^m$, then Single Layer Feedforward Network (SLFN) with $N$ Hidden nodes is designed using the following equation:

$$\sum_{i=1}^{H} \beta_i f(w_i x_j + b_i), j \in [1, N]$$

(3.12)

where $f$ = activation function, $w_i$ = input weights, $b_i$ = bias and $\beta_i$ = output weights.

The error between the target output and actual output is given by the following equation:

$$\sum_{i=1}^{H} \beta_i f(w_i x_j + b_i) = t_j, j \in [1, N]$$

(3.13)

where $t_j$ is the actual output.

The above equation can be simply written as follows:

$$\beta = H^T Y.$$ 

(3.14)

where

$$\beta = (\beta_1^T \ldots \beta_H^T)^T$$

(3.15)

$$Y = (y_1^T \ldots y_N^T)^T$$

(3.16)

and $H$ = the Hidden layer output matrix and is calculated using the following equation:

$$H = \begin{pmatrix}
  f(w_1 x_1 + b_1) & \ldots & f(w_H x_1 + b_H) \\
  \vdots & \ddots & \vdots \\
  f(w_1 x_N + b_1) & \ldots & f(w_H x_N + b_H)
\end{pmatrix}$$

(3.17)
The hidden layer output matrix H is determined by randomly initializing the weights \((w_i)\) and the bias \((b_i)\).

The output weights \(\beta\) are calculated from the knowledge of the hidden layer output matrix H and target values \(Y\) using a Moore-Penrose generalized inverse of the matrix H, which is denoted as \(H^\dagger\). Table 3.4 summarizes the ELM Algorithm [41, 42].

**Table 3.4 Extreme Learning Machine Algorithm**

<table>
<thead>
<tr>
<th>Given a training set (\mathbf{X} = (x_i, t_i)), (x_i \in R^n), (t_i \in R^m), an activation function (f: R \mapsto R) and the number of hidden nodes (H).</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Randomly assign input weights (w_i) and bias (b_i), (i \in [1, H]).</td>
</tr>
<tr>
<td>- Calculate the hidden layer output matrix (H).</td>
</tr>
<tr>
<td>- Calculate output weight matrix (\beta = H^\dagger Y).</td>
</tr>
</tbody>
</table>

### 3.10.2. Particle Swarm Optimization - Extreme Learning machine (PSO-ELM) Wrapper approach

Particle Swarm Optimization (PSO) [9, 45, 47 and 71] is a stochastic search technique that aims to optimize an objective function, motivated by social activities of birds gathering or fish schooling. PSO is a population dependent search algorithm in which each individual is indicated as a particle and represents a candidate solution. All the particles in PSO moves through the search space with an adjustable velocity that is dynamically transformed based on its individual moving experience and also the moving experience of the other particles. In PSO each particles attempt to enhance themselves by imitating traits from their successful peers. The position in proportion to the best fitness is known as pbest (Particle best) and the overall best out of all the particles in the population is called
gbest (Global best) [71]. The velocity $v_i(t)$ and the positions $x_i(t)$ of the particles are updated with the following equations.

$$v_i(t + 1) = w \cdot v_i(t) + c_1 r_1 (y_i(t) - x_i(t)) + c_2 r_2 (\hat{y}(t) - x_i(t))$$  \hspace{1cm} (3.18)$$

$$x_i(t + 1) = x_i(t) + v_i(t + 1)$$  \hspace{1cm} (3.19)$$

where $i=1, 2, ..., n$

$v_i(t)$: Velocity of agent $i$ at iteration $t$ and must lie in the range

$$V_{min} \leq V_i(t) \leq V_{max}$$  \hspace{1cm} (3.20)$$

$w$: inertia weight factor

c$_1$, c$_2$: Cognitive and social acceleration factors respectively,

$r_1, r_2$: Uniformly distributed random number between 0 and 1,

$x_i(t)$: Current position of agent $i$ at iteration $t$,

$y_i(t)$: Personal best (pbest) and is updated using the following equation:

$$y_i(t + 1) = \begin{cases} 
  y_i(t), & f(x_i(t + 1)) \geq f(y_i(t)) \\
  x_i(t + 1), & f(x_i(t + 1)) < f(y_i(t)) 
\end{cases}$$  \hspace{1cm} (3.21)$$

$\hat{y}(t)$: Global best (gbest) and is updated using the following equation:

$$\hat{y}_i(t + 1) = \arg\min_i \{ f(y_i(t + 1)) \}, i \leq i \leq s$$  \hspace{1cm} (3.22)$$

where $s$ = number of particles in the swarm.

The general flowchart of Particle Swarm Optimization algorithm is given in the figure 3.10:
In the proposed PSO-ELM Wrapper approach, Duration, Latency, Trigraph, Digraph and the proposed Virtual Key Force are given as input features. Feature subset selection is done and the selected features are evaluated by Extreme Learning Machine in order to evaluate the fitness of features. The process is repeated until the best solution is obtained. Flow chart for proposed PSO-ELM wrapper approach for feature subset selection is given in the following figure 3.11. The fitness function of the PSO is evaluated using ELM.
The algorithm of the proposed PSO-ELM Wrapper approach is given in the following table 3.5.

**Table 3.5 Proposed PSO-ELM Wrapper Approach Algorithm**

*Input:* Duration, Latency, Trigraph, Digraph, Virtual Key Force  
*Output:* Subset feature values.  
**Step 1:** Initialize the number of Iterations, Number of particles, Weight, $c_1, c_2, r_1, r_2$, $v_i(t)$.  
**Step 2:** Compute the feature values of $x_i(t)$ (Duration, Latency, Digraph, Trigraph, and Virtual Key Force).  
**Step 3:** Evaluate fitness for each feature value using $ELM$.  
**Step 4:** The following is repeated for number of iterations:  
1. Check if $p$ $>$ $p_{best}$ then, $p_{best} = p$ else $p_{best} = p_{best}$.  
2. If $p_{best} \geq g_{best}$ then, $g_{best} = p_{best}$ else $g_{best} = g_{best}$.  
3. Update velocity by (3.18) and position is updated by (3.19), $p_{best}$ and $g_{best}$ position are updated using (3.21) and (3.22).  
**Step 4** is repeated until $g_{best}$ is optimum value.

Genetic Algorithm (GA) [44, 71, 80, and 84] is a stochastic search technique based on natural selection of population. In Genetic Algorithm, the fitness of all individuals in the population is calculated in every generation by choosing several individuals from the existing population according to their fitness value, and updated to generate a new population. Genetic algorithms combine selection, crossover, and mutation operators in order to find the best solution to a problem. Chromosomes are selected from the population to be parents to crossover. The selection is done based on the following types which are given in the table 3.6.

**Table 3.6 Chromosome Selection Types**

<table>
<thead>
<tr>
<th>Selection Types</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roulette Wheel</td>
<td>The chance of a chromosome getting selected is proportional to its fitness (or rank)</td>
</tr>
<tr>
<td>Tournament</td>
<td>Uses roulette selection N times to produce a tournament subset of chromosomes</td>
</tr>
<tr>
<td>Top Percent</td>
<td>Randomly selects a chromosome from the top N percent of the population as specified by the user</td>
</tr>
<tr>
<td>Best</td>
<td>Selects the best chromosome determined by fitness</td>
</tr>
<tr>
<td>Random Selection</td>
<td>Randomly selects a chromosome from the population</td>
</tr>
</tbody>
</table>

A new chromosome called Offspring is produced by Crossover operator which combines two chromosomes or parents. The following types of crossover which is shown in the table 3.7 are available.
Table 3.7 Crossover Types

<table>
<thead>
<tr>
<th>Crossover Types</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Point</td>
<td>Randomly selects a crossover point within a chromosome then interchanges two parent chromosomes to produce two new offspring</td>
</tr>
<tr>
<td>Two Point</td>
<td>Randomly selects two crossover points within a chromosome then interchanges two parent chromosomes to produce two new offspring</td>
</tr>
<tr>
<td>Uniform</td>
<td>Decides (by the mixing ratio) which parent will contribute each of the gene values in the offspring chromosomes</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>Linearly combines two parent chromosome vectors to produce two new offspring</td>
</tr>
<tr>
<td>Heuristic</td>
<td>Uses the fitness values of the two parent chromosomes to determine the direction of the search</td>
</tr>
</tbody>
</table>

Mutation alters one or more gene values in a chromosome from its initial state and result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution. The following table 3.8 shows the different types of Mutation operator available.

Table 3.8 Mutation Types

<table>
<thead>
<tr>
<th>Mutation Types</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip Bit</td>
<td>Inverts the value of the chosen gene</td>
</tr>
<tr>
<td>Boundary</td>
<td>Replaces the value of the chosen gene randomly with either the upper or lower bound for that gene</td>
</tr>
</tbody>
</table>
Uniform increases the probability such that the amount of the mutation will be close to 0 as the generation number increases.

Non Uniform replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene.

Gaussian adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene.

The flowchart of general Genetic Algorithm is given in the figure 3.12.

![Flowchart of Genetic Algorithm](image)

**Figure 3.12 Flowchart of Genetic Algorithm**

The flow chart of the proposed GA-ELM wrapper approach for feature subset selection is given in the following figure 3.13. The fitness function of the GA is evaluated using ELM-ANP.
Figure 3.13 Flowchart of Proposed GA-ELM Wrapper Feature Subset Selection

In the proposed GA-ELM wrapper approach, duration, latency, trigraph, digraph and the proposed virtual key force are given as input features. Feature subset selection is done after crossover and mutation and selected features are evaluated by ELM to find fitness value. The process is repeated until best solution is obtained. The algorithm of GA-ELM wrapper approach is given in table 3.9.

Table 3.9 Proposed GA-ELM Wrapper Approach Algorithm

| Input: Duration, Latency, Trigraph, Digraph, Virtual Key Force |
| Output: Subset feature values. |
| Step 1: Extract the feature values Duration, Latency, Trigraph, Digraph and Virtual Key Force. |
| Step 2: Initialize Number of generations, Initial population, Crossover rate and mutation rate. |
| Step 3: Generate pool of candidate feature subset. |
| Step 4: Perform crossover and mutation operations and generate new pool of candidate feature subset. |
| Step 5: Evaluate the selected subset by ELM. |
| Step 6: Selected feature subset by ELM are put back into the population. |
| Step 7: Repeat Steps 3 to 6 until the best solution or maximum iteration is reached. |
| The best solutions are achieved in the end. |
3.10.4. Ant Colony Optimization - Extreme Learning Machine (ACO-ELM)

Wrapper approach

Ant Colony Optimization (ACO) [46] algorithms have been introduced based on the observation of the real ant colonies. A unique characteristic of ant colonies is their foraging behavior. The ability of the ants to find the shortest route between their nest and a food source is a very significant character of the ant colonies. Ant colony optimization algorithms have been used to provide better solutions to many search problems.

Figure 3.14 [35] shows the behavior of the ants in finding the shortest distance between the nest and the food. The ants that find the closest food source leave more amounts of pheromone substance as a mark for the other ants. The ants take the route in which the pheromone level is stronger. Thus more ants choose to take the shorter path, the level of pheromone increases until every ant is taking the same shorter path.

![Figure 3.14 Ants Converging on the Shortest Path between a Food Source and the Nest](image)

The flowchart of basic ACO algorithm is given in the following figure 3.15.
A wrapper based Ant Colony optimization with ELM has been proposed in this thesis for feature subset selection. The Flow chart of proposed ACO-ELM wrapper approach for feature subset selection is given in the following figure 3.16. The fitness function of the ACO is evaluated using ELM.

In the proposed ACO-ELM Wrapper approach, Duration, Latency, Trigraph, Digraph and the proposed Virtual Key Force are given as input features. Feature subset selection is done and the selected features are evaluated by Extreme
Learning Machine in order to evaluate the fitness of features. The process is repeated until the best solution is obtained. The algorithm of ACO-ELM Wrapper approach is given in the following table 3.10.

**Table 3.10 Proposed ACO - ELM Wrapper Approach Algorithm**

| **Input:** | Duration, Latency, Trigraph, Digraph, Virtual Key Force |
| **Output:** | Subset feature values. |
| **Step 1:** | Extract the feature values Duration, Latency, Trigraph, Digraph and Virtual Key Force. |
| **Step 2:** | Initialize Number of iterations, Number of Ants, Initial pheromone value associated with each feature and pheromone evaporation rate. |
| **Step 3:** | Select a random feature value for each ant with the criteria that the particular feature value should not have been selected previously by the ant and generate a subset for each Ant. |
| **Step 4:** | Evaluate the selected subset of each ant by ELM. |
| **Step 5:** | Exit, if the number of iterations is more than the maximum number of iterations, otherwise continue. |
| **Step 6:** | Update the pheromone value for features which are selected in step 4. |
| **Step 7:** | Generate new ants by removing the old ants. |
| **Step 8:** | Repeat steps 3 to 7 until the last iteration. |
| **Step 9:** | Set best feature subset as global best. |

### 3.11. CONCLUSION OF PHASE III

PSO, GA and ACO wrapped with ELM algorithm is used for feature subset selection. Experimental results compare ACO-ELM, PSO-ELM and GA-ELM wrapper approaches and revealed that the wrapper based on ACO-ELM approach is the best method for feature subset selection, since the number of features selected is very low compared with other methods.

The next section discusses the classification methods. There are various classification techniques available in the literature. The most widely used classification approaches are Artificial Neural Networks such as BPN. The
following section describes the role of neural network techniques in classification approaches.

3.12. CLASSIFICATION

The best class that is closest to the classified pattern is found using Classification. Neural Networks are one of the most widely used techniques for classification. Neural networks are a computational model motivated by the connectivity of neurons in living nervous systems. Artificial Neural Network (ANN) is observed to be very useful in classification approaches with better accuracy and performance. ANN is a data processing model that is motivated by the manner of certain biological nervous system like the brain that processes the data.

3.12.1. Back Propagation Neural Network (BPN)

Artificial Neural Networks (ANN) comprises of a huge number of interrelated processing neurons operating together to resolve particular issues. ANN has been significantly applied for various applications in many fields. Among ANN, Back Propagation Neural Network (BPN) is observed to provide the best results in terms of accuracy. It often facilitates fast convergence [2] on acceptable local minima for error in the type of networks to which it is suited. The Architecture of Back Propagation Neural Network is given in the figure 3.17.
Figure 3.17 Back-Propagation Neural Network Architecture

The input vectors $x_1$ to $x_d$ are given as input to the input layers $I_1$ to $I_d$. Weights are randomly assigned between input and hidden layers and the input vectors are multiplied with the assigned weights and given as input to hidden layers $H_1$ to $H_e$. The outputs of hidden layers are calculated using sigmoid function. Weights are randomly assigned between hidden and output layers and the output of hidden layer are multiplied with the assigned weights and given as input to output layers $O_1$ to $O_c$. The outputs of output layers are calculated using sigmoid function. Error (delta value) is found by subtracting the output from the target output. Weights are adjusted to achieve the target output. The adjusted weight is multiplied with the delta value and given to the hidden layers. Weights on each neuron of hidden layer are adjusted and given as input to the input layer. The above process is repeated until the target output is equal to the desired output.

The number of neurons in the input layer is found out by the available number of input. The number of neurons in the output layer is determined by the number of preferred output. The number of hidden layers and the number of neurons in the hidden layers cannot be defined prior. However, the addition of a hidden layer may facilitate the network to be trained on more complex patterns, but at the equivalent time it reduces its performance.
There are basically two types of learning namely, on-line learning and batch learning. The propagation is followed directly with the help of a weight update in online learning. In batch learning, much propagation happens before weight updating is carried out. Batch learning requires extra memory capacity, but on-line learning needs more updates.

The Back Propagation Neural Network algorithm is given in the table 3.11.

**Table 3.11 Back Propagation Neural Network Algorithm**

| Input: Duration, Latency, Trigraph, Digraph, Virtual Key Force |
| Output: Classification accuracy. |
| Step 1: Assign the feature values to input neurons. |
| Step 2: Initialize the weights randomly. |
| Step 3: Calculate the input to the hidden layers by multiplying the weight with input value given to input neuron using Sigmoid function. |
| Step 4: Calculate the input to the output layers by multiplying the weight with the output value of hidden neuron. |
| Step 5: Calculate the output from output neurons using Sigmoid function. |
| Step 6: Calculate the delta value by subtracting the output from target value. |
| Step 7: Update the hidden to output layer weights. |
| Step 8: Update the input to hidden layer weights. |
| Step 9: Repeat steps 3 to 8 until the target output is equal to the desired output. |

**3.13. EXTREME LEARNING MACHINE (ELM)**

Computational intelligence techniques are widely being used in variety of applications. Neural Networks like Back Propagation Neural Network (BPN) and Support Vector Machines (SVMs) are the most popular computational intelligence
techniques that have been playing the leading roles. However, both neural networks and SVMs face some challenging issues like trivial human intervene slow learning speed and poor computational scalability. The parameters of feedforward network such as BPN are manually tuned or altered and thus there lies the dependency between different layers of parameters (weights and bias) [7]. Gradient descent-based techniques have been widely used as learning algorithms of feedforward neural networks. However, gradient descent-based learning techniques are usually very slow because of their inappropriate learning steps or may easily converge to local minima and various iterative learning steps may be needed by such learning algorithms in order to obtain better learning performance.

Extreme learning machine (ELM), an emerging technology proposed by [41, 42] overcomes the challenges faced by the above mentioned computational intelligent techniques. ELM works for Single Hidden Layer Feedforward Networks (SLFNs). The hidden layer of SLFNs need not be tuned which is an advantage of Extreme Learning Machine. In SLFN, the input weights and hidden layer bias can be randomly assigned. After the input weights and the hidden layer bias are selected at random, SLFNs can be regarded as a linear system and the output weights which link the hidden layer to the output layer of SLFNs can be systematically found out with simple generalized inverse operation (Moore-Penrose generalized inverse) of the hidden layer output matrices. The learning speed of ELM is faster and with least human intervene it gives better performance when compared with other computational intelligent techniques.

Hence Extreme Learning algorithm discussed in section 3.10.1 is adapted for classification of keystroke features. The training algorithm for ELM is Levenberg-Marquardt (LM). The wrapper based classification technique is proposed in the thesis. The features selected by feature subset selection methods are given as input to input nodes of ELM for classification.
3.14. MODIFIED LEVENBERG-MARQUARDT ALGORITHM

The learning parameter $\mu$ is a constant number in standard Levenberg-Marquardt algorithm which is modified as following using modified LM algorithm [5].

$$
= 0.01 e^T.e
$$

(3.23)

where $e^T$ = error vector

The Modified LM algorithm [7] is given below:

Step 1: Initialize the weights.

Step 2: Present all input to the network and calculate the corresponding network output and errors $e_q$ and compute the sum of squared errors over all inputs.

$$
e_q = t_q - a_q^m
$$

(3.24)

where $e_q$ = training error at output, $t_q$ = target output vector, $a_q^m$ = actual output vector

Step 3: Compute the Jacobian matrix $J$ as follows:

Compute the sensitivities with the recurrence relations $S^m_q$, after initializing $S^m_q$

$$
S^m_q = f^m(n^m_q)(w^{m+1})^T. S^{m+1}_q
$$

(3.25)

$$
S^M_q = -f^m(n^m_q)
$$

(3.26)

where $f^m(n^m_q)$ = matrix obtained using Gauss Newton method

Augment the individual matrices into the Marquardt sensitivities using the following equation.

$$
S^m = [S^m_1, S^m_2, ..., S^m_q]
$$

(3.27)

Calculate the elements of the Jacobian matrix with the equations

$$
[J]_{h,l} = S^m_{i,h} \times S^{m-1}_{j,k}
$$

(3.28)

where $S^m_{i,h}, S^{m-1}_{j,k}$ are elements of Jacobian Matrix

and

$$
[J]_{h,l} = S^m_{i,h}
$$

(3.29)
Step 4: Determine increments of weights, $\Delta w_k$ by the following equation

$$\Delta w_k = [J^T(w_k).J(w_k) + .1]^{-1}.J^T(w_k).e(w_k)$$

(3.30)

where (learning parameter) $= 0.01$ $e^T e$, $I$ = Identity matrix, $J$= Jacobian Matrix, $W$= Weight vectors

Step 5: The sum of the squared errors is recomputed using $w + \Delta w$. If this new sum of squared errors is lesser than the sum of squared errors calculated in step 2, then

$$= \beta (\beta = .1)$$

(3.31)

where $\beta$= decay rate

$$w_{k+1} = w_k + \Delta w_k$$

(3.32)

and continue from step 4.

otherwise compute with the following equation and continue step 5.

$$= \bar{\beta}$$

(3.33)

3.15. ANALYTIC NETWORK PROCESS (ANP)

The input weights and the hidden bias are randomly chosen in Extreme Learning machine. The output weights are analytically determined by Moore Penrose inverse. ELM overcomes the problems such as slow learning speed, number of epochs etc., which are being faced by the computational intelligence techniques such as BPN and SVMs and has faster learning rate. However, ELM requires more number of hidden neurons than that of human intervene based algorithms that are mentioned above. The input weights are generated using Analytic Network Process (ANP).

ANP proposed by [66] facilitates for more complex interrelationships among the decision levels and attributes and does not require any strict hierarchical structure. The ANP approach handles interdependent relationships among the elements by obtaining the composite weights developed by the
supermatrix. The supermatrix concept contains parallels to the Markov chain process where relative importance weights are adjusted by forming a supermatrix from the eigenvectors of these relative importance weights. The weights are adjusted by determining products of the supermatrix. The mathematical derivation of ANP algorithm has been presented by [7]. The supermatrix formation is given in the following figure 3.18.

\[
W_k = \begin{pmatrix}
  e_{11} & e_{12} & \ldots & e_{1B_1} & W_{11} & W_{12} & \ldots & W_{1N} \\
  e_{12} & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  e_{1B_1} & e_{21} & \ldots & W_{21} & W_{22} & \ldots & W_{2N} \\
  \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  e_{N1} & e_{N2} & \ldots & W_{N1} & W_{N2} & \ldots & W_{NN} \\
  e_{N2} & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  e_{NB_N} & e_{NB_N} & \ldots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\end{pmatrix}
\]

**Figure 3.18 Supermatrix**

The weights obtained from the supermatrix are used as input weights of Extreme Learning Machine.

### 3.16. PROPOSEDWRAPPER BASED CLASSIFICATION WITH ELM - ANP APPROACH – PHASE IV

A wrapper based classification is proposed in the thesis. Feature subset selection is done for the original extracted features using wrapper based approach in which the performance is evaluated using the classifier. The selected feature set acts as training set and testing set (70% as training set and 30% as test set). Classification is done using ELM and ELM-ANP algorithms and the final
accuracy of the system is estimated. The wrapper based classification approach is shown in the figure 3.19.

![Wrapper Based Classification Approach](image)

**Figure 3.19 Wrapper Based Classification Approach**

In the proposed wrapper based classification approach, the features are selected by feature subset selection methods such as PSO, GA and ACO and are given as input vectors to the input layers. The input weights and hidden bias are generated by Analytic Network Process. Extreme Learning Machine analytically determines the output weights and the weights and bias are updated using Modified Levenberg–Marquardt (LM) algorithm which is used to train the network.

The process for the Extreme Learning Machine with ANP is described below:

Given a training set \( \mathbf{X} = (x_i, t_i), x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m \), an activation function \( f_1(x) \) and \( f_2(x) \), \( N \) & \( K \), the number of nodes hidden in first and second layer.
Step 1: The input weight vectors $w_1$ are randomly initialized with the help of ANP technique and bias vectors without using the ANP technique.

Step 2: Compute the hidden first layer output matrix $a_1$. Using ELM algorithm, determine the output weight.

$$w_2 = a_1^{-1} \cdot t$$  \hspace{1cm} (3.34)

Step 3: Determine the hidden second layer output matrix $a_2$ errors

$$e_1 = t - a_2$$  \hspace{1cm} (3.35)

and determine the sum of squared errors of overall input.

Step 4: Compute the Jacobian matrix $J$ as follows:

Computer the sensitivities with the recurrence relations $S^m_q$, after initializing $S^m_q$ using the equations (3.25) and (3.26).

Augment the individual matrices into the Marquardt sensitivities using the equation (3.27).

Calculate the elements of the Jacobian matrix with the equations (3.28) and (3.29).

Step 5: Determine increments of weights, $\Delta w_k$ by (3.30)

Step 6: Update weight vectors and bias.

Step 7: Recompute the sum of the squared errors using $w + \Delta w$. If this new sum of squared is lesser than the sum of squared errors calculated in step 3, then calculate $\mu$ with (3.31) and $w_{k+1}$ with (3.32) and continue from step 5. Otherwise compute $\mu$ with (3.33) and continue step 7.
Thus, by using the ANP technique, the Extreme Learning Machine is modified as ELM-ANP which has the advantage of quick learning.

3.17. CONCLUSION OF PHASE IV

This section focused on the ELM classification based on ANP approach. ANP is utilized in finding the weights. The proposed methodology provides good classification accuracy and considerably reduces the training time and testing time. The generalization performance of the ELM is clearly explained in this chapter which enhances the performance of the whole system.

3.18. PROPOSED WRAPPER BASED ELM – ANP FOR FEATURE SUBSET SELECTION

The proposed ELM – ANP classification algorithm is applied as wrapper based feature subset selection approach with PSO, GA and ACO. The obtained experimental results show that wrapper based approach using ELM-ANP still reduces more number of features than the proposed ELM wrapper approach.


Flow chart for PSO-ELM-ANP wrapper approach for feature subset selection is given in the following figure 3.20. The fitness function of the PSO is evaluated using ELM-ANP.
Figure 3.20 Flowchart of Proposed PSO-ELM-ANP Wrapper Feature Subset Selection

In the proposed PSO-ELM-ANP Wrapper approach, Duration, Latency, Trigraph, Digraph and the proposed Virtual Key Force are given as input features. Feature subset selection is done and the selected features are evaluated by Extreme Learning Machine- Analytic Network Process. The process is repeated until the best solution is obtained.

The algorithm of PSO-ELM-ANP Wrapper approach is given in the following table 3.12.

Table 3.12 Proposed PSO – ELM – ANP Wrapper Approach Algorithm

| Input: Duration, Latency, Trigraph, Digraph, Virtual Key Force |
| Output: Subset feature values. |
| Step 1: Initialize the number of Iterations, Number of particles, Weight, $c_1$, $c_2$, $r_1$, $r_2$, $v_i(t)$. |
| Step 2: Compute the feature values of $x_i(t)$ (Duration, Latency, Digraph, Trigraph, and Virtual Key Force). |
| Step 3: Evaluate fitness for each feature value using ELM-ANP. |
| Step 4: The following is repeated for number of iterations: |
| 1. Check if $p$ >= $p_{best}$ then, $p_{best}$ = $p$ else $p_{best}$ = $p_{best}$. |
| 2. If $p_{best}$ >= $g_{best}$ then, $g_{best}$ = $p_{best}$ else $g_{best}$ = $g_{best}$. |
| 3. Update velocity by (3.18) and position is updated by (3.19), $p_{best}$ and $g_{best}$ position are updated using (3.21) and (3.22). |
| Step 4 is repeated until $g_{best}$ is optimum value. |

The flow chart for GA-ELM wrapper approach for feature subset selection is given in the following figure 3.21. The fitness function of the GA is evaluated using ELM-ANP.

**Figure 3.21 Flowchart of Proposed GA-ELM-ANP Wrapper Feature Subset Selection**

In the proposed GA-ELM-ANP Wrapper approach, Duration, Latency, Trigraph, Digraph and the proposed Virtual Key Force are given as input features. Feature subset selection is done after Crossover and Mutation and the selected features are evaluated by Extreme Learning Machine – Analytic Network Process. The process is repeated until the best solution is obtained. The algorithm of GA-ELM-ANP Wrapper approach is given in the following table 3.13.
Table 3.13 Proposed GA - ELM –ANP Wrapper Approach Algorithm

| **Input:** Duration, Latency, Trigraph, Digraph, Virtual Key Force  |
| **Output:** Subset feature values.  |
| **Step 1:** Extract the feature values Duration, Latency, Trigraph, Digraph and Virtual Key Force.  |
| **Step 2:** Initialize Number of generations, Initial population, Crossover rate and mutation rate.  |
| **Step 3:** Generate pool of candidate feature subset.  |
| **Step 4:** Perform crossover and mutation operations and generate new pool of candidate feature subset.  |
| **Step 5:** Evaluate the selected subset by ELM-ANP.  |
| **Step 6:** Selected feature subset by ELM-ANP are put back into the population.  |
| **Step 7:** Repeat Steps 3 to 6 until the best solution or maximum iteration is reached.  |
| The best solutions are achieved in the end.  |


The flow chart for ACO-ELM-ANP wrapper approach for feature subset selection is given in the following figure 3.22. The fitness function of the ACO is evaluated using ELM-ANP.
In the proposed ACO-ELM-ANP Wrapper approach, Duration, Latency, Trigraph, Digraph and the proposed Virtual Key Force are given as input features. Feature subset selection is done and the selected features are evaluated by ELM-ANP in order to find the fitness value. The process is repeated until the best solution is obtained. The algorithm of ACO-ELM-ANP Wrapper approach is given in the following table 3.14.

**Table 3.14 Proposed ACO – ELM –ANP Wrapper Approach Algorithm**

| Input: Duration, Latency, Trigraph, Digraph, Virtual Key Force |
| Output: Subset feature values. |
| **Step 1**: Extract the feature values Duration, Latency, Trigraph, Digraph and Virtual Key Force. |
| **Step 2**: Initialize Number of iterations, Number of Ants, Initial pheromone value associated with each feature and pheromone evaporation rate. |
| **Step 3**: Select a random feature value for each ant with the criteria that the particular feature value should not have been selected previously by the ant and generate a subset for each Ant. |
| **Step 4**: Evaluate the selected subset of each ant by ELM-ANP. |
| **Step 5**: Exit, if the number of iterations is more than the maximum number of iterations, otherwise continue. |
| **Step 6**: Update the pheromone value for features which are selected in step 4. |
| **Step 7**: Generate new ants by removing the old ants. |
| **Step 8**: Repeat steps 3 to 7 until the last iteration. |
| **Step 9**: Set best feature subset as global best. |
3.19. CONCLUSION OF ELM-ANP WRAPPER BASED FEATURE SUBSET SELECTION

PSO, GA and ACO wrapped with ELM-ANP algorithm is used for feature subset selection. Experimental results compare PSO, GA and ACO wrapped with ELM-ANP respectively and revealed that the wrapper based ACO-ELM-ANP approach is the best method for feature subset selection, since the number of features selected is very low when compared with other methods. The next chapter discusses the results obtained using proposed methods in detail.