# Data Mining Feature Subset Weighting and Selection Using Genetic Algorithms

Okan Yilmaz

# DATA MINING FEATURE SUBSET WEIGHTING AND SELECTION USING GENETIC ALGORITHMS

THESIS

Okan Yilmaz, Lieutenant, TUAF

AFIT/GCE/ENG/02M-05

**DEPARTMENT OF THE AIR FORCE**

**AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

**Wright-Patterson Air Force Base, Ohio**

# Report Documentation Page

| | | |
|---|---|---|
| **Report Date**<br>18 Mar 02 | **Report Type**<br>Final | **Dates Covered (from... to)**<br>Jun 2001 - Mar 2002 |

| | |
|---|---|
| **Title and Subtitle**<br>Data Mining Feature Subset Weighting and Selection Using Genetic Algorithms | **Contract Number** |
| | **Grant Number** |
| | **Program Element Number** |
| **Author(s)**<br>Lt Okan Yilmaz, TUAF | **Project Number** |
| | **Task Number** |
| | **Work Unit Number** |
| **Performing Organization Name(s) and Address(es)**<br>Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Bldg 640 WPAFB OH 45433-7765 | **Performing Organization Report Number**<br>AFIT/GCE/ENG/02M-05 |
| **Sponsoring/Monitoring Agency Name(s) and Address(es)**<br>Dr. Rober L. Ewing, Tech Adv. AFRL/IFTA WPAFB OH 45433 | **Sponsor/Monitor's Acronym(s)** |
| | **Sponsor/Monitor's Report Number(s)** |
| **Distribution/Availability Statement**<br>Approved for public release, distribution unlimited | |
| **Supplementary Notes**<br>The original document contains color images. | |

**Abstract**
We present a simple genetic algorithm (sGA), which is developed under Genetic Rule and Classifier Construction Environment (GRaCCE) to solve feature subset selection and weighting problem to have better classification accuracy on k-nearest neighborhood (KNN) algorithm. Our hypotheses are that weighting the features will affect the performance of the KNN algorithm and will cause better classification accuracy rate than that of binary classification. The weighted-sGA algorithm uses real-value chromosomes to find the weights for features and binary-sGA uses integer-value chromosomes to select the subset of features from original feature set. A Repair algorithm is developed for weighted-sGA algorithm to guarantee the feasibility of chromosomes. By feasibility we mean that the sum of values of each gene in a chromosome must be equal to 1. To calculate the fitness values for each chromosome in the population, we use K Nearest Neighbor Algorithm (KNN) as our fitness function. The Euclidean distance from one individual to other individuals is calculated on the d-dimensional feature space to classify an unknown instance. GRaCCE searches for good feature subsets and their associated weights. These feature weights are then multiplied with normalized feature values and these new values are used to calculate the distance between features.

**Subject Terms**
Data Mining, Genetic Algorithm, Classification, Feature Subset Weighting and Selection, K Nearest Neighbor Algorithm, Feasibility of a Chromosome, Repair Algorithm, and Fitness Landscape.

| | |
|---|---|
| **Report Classification**<br>unclassified | **Classification of this page**<br>unclassified |
| **Classification of Abstract**<br>unclassified | **Limitation of Abstract**<br>UU |
| **Number of Pages**<br>124 | |

AFIT/GCE/ENG/02M-05

# Data mining Feature Subset Weighting and Selection Using Genetic Algorithms

Presented to the faculty of the Graduate School of Engineering & Management

Of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Computer Engineering)

Okan Yilmaz, B. S.

Lieutenant, TUAF

March 2002

.

AFIT/GCE/ENG/02M-05

# DATA MINING FEATURE SUBSET WEIGHTING AND SELECTION USING GENETIC ALGORITHMS

THESIS

Okan Yilmaz, B. S.
Lieutenant, TUAF

Approved:

| | |
|---|---|
| [illegible] | 15 MARCH '02 |
| Dr. Gary B. Lamont (Chairman) | date |
| [illegible] | 18 Mar 02 |
| Dr. Richard Raines (Member) | date |
| [illegible] | 18 MARCH 02 |
| Major Karl. M. Mathias (Member) | date |

# ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my faculty advisor, Dr. Gary B. Lamont, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated.

I am also indebted to the many maintenance professionals who spent their valuable time explaining the processes and procedures they used in the maintenance of their support equipment.

My deepest appreciation and thanks go to the researchers who developed GRaCCE from the scratch and who worked on it and improved it’s performance in many ways in following years and left a powerful application to solve real-world problems at hand.

I would not have succeeded if my family did not support me within the period of this enduring effort. I would like to express my grateful appreciation to my parents, my wife, my brother, and my son. Especially my wife and my son shared my sleepless nights and restless weekends and graciously put up many relentless studying days with me and gave me the strength and endurance I needed to finish my education in AFIT. Without their support, this journey would have been a whole lot tougher.

The real heroes who deserve my greatest appreciation are the citizens of my country who helped me earn this great education. I express my sincere appreciation to each citizen of Turkey.

Okan Yilmaz

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

We present a simple genetic algorithm (sGA), which is developed under Genetic Rule and Classifier Construction Environment (GRaCCE) to solve feature subset selection and weighting problem to have better classification accuracy on k-nearest neighborhood (KNN) algorithm. Our hypotheses are that weighting the features will affect the performance of the KNN algorithm and will cause better classification accuracy rate than that of binary classification. The weighted-sGA algorithm uses real-value chromosomes to find the weights for features and binary-sGA uses integer-value chromosomes to select the subset of features from original feature set. Since we use real-value chromosomes for weighted-sGA, instead of using standard crossover and mutation operators, these GRaCCE sGA operators are modified to adjust them to the feature subset selection and weighting problem. Linear ranking selection method is used for breeding new individuals and two-point crossover operator is used for both algorithms. Exchange mutation operator is selected for weighted-sGA and inversion mutation operator is selected for binary-GA. A Repair algorithm is developed for weighted-sGA algorithm to guarantee the feasibility of chromosomes. By feasibility we mean that the sum of values of each gene in a chromosome must be equal to 1. To calculate the fitness values for each chromosome in the population, we use K Nearest Neighbor Algorithm (KNN) as our fitness function. The Euclidean distance from one individual to other individuals is calculated on the d-dimensional feature space to classify an unknown instance. GRaCCE searches for good feature subsets and their associated weights. These feature weights are then multiplied with normalized feature values and these new values are used to calculate the distance between features.

The previous GRaCCE algorithm sorts the calculated distances to find the k nearest neighbors to an unknown instance but this is a huge bottleneck in the GraCCE algorithm since the complexity of sort operation is O ($n^2$) and this sorting has to be done for each Eucidean distance calculation. We know that K for KNN algorithm is usually a very small integer number ($K<20$). So that instead of sorting, the new GRaCCE algorithm searches only for the K minimum distance values in the neighborhood list matrix.

The results show that by giving appropriate weights to features it is possible to have better classification accuracy than that of binary feature subset selection.

# DATA MINING FEATURE SUBSET WEIGHTING AND SELECTION USING GENETIC ALGORITHMS

## *I. Introduction*

The widespread availability of relatively inexpensive but powerful computing has led to another revolution, the storage of massive amounts of data in electronic form. Raw data stored in data warehouses are not useful for decision making unless necessary data mining rules are inducted from these raw data. Data mining is the process of searching for these rules, which then are presented to decision makers to be employed for other real world data. The concept of data mining was started almost 10 years ago. The interest in data mining field and its exploitation in different domains (marketing, finances, banking, engineering, health care, power systems, meteorology, military, etc.) has increased recently due to a combination of the following factors:

Emergence of very large amounts of data (terabytes of data due to computer automated data measurement and/or collection, digital recording, centralized data archives, and software and hardware simulations).

Dramatic cost decrease of mass storage devices

Emergence and quick growth of fielded database management systems

Advances in computer technology such as faster computers and parallel architectures

Continuous developments in automatic learning techniques

Possible presence of uncertainty in data (noise outliers, missing information)

The general purpose of data mining is to process the information from the enormous stock of data we have or that we may generate, so as to develop better ways to handle data and support future decision-making. Sometimes, the patterns to be searched for and the models to be extracted from data are subtle and require complex calculus and/ or significant specific domain knowledge. Or even worse, there are situation where one would like to search for patterns that humans are not well suited to find, even if they are experts in the field. For example, in many power system related problems, one is faced with high dimensional data sets that can not be easily modeled and controlled on the whole, and, therefore, automatic methods capable of synthesizing structures from such data become a necessity.

## 1.1 Background

This section provides background information needed to understand the thesis problem. This information includes material on data mining, classification, feature subset selection and weighting problem and genetic algorithms.

### 1.1.1 What is Data Mining?

Data mining is the search for valuable information in large volumes of data. There are many other terms carrying a similar or slightly different meaning to data mining, such as *knowledge mining from databases, knowledge extraction, data/pattern analysis, data archaeology, and data dredging.* Many people treat data mining as a synonym for another popularly used term, “Knowledge Discovery in Databases”, or KDD [1]. Data mining is the process of finding patterns and relations in large databases [21]. The primary purpose of the data mining is to extract information from huge amounts of raw data [22]. Data mining using statistical methods [23] have been quite successful.

Traditional data analysis is assumption driven in the sense that a hypothesis is manually formed and validated (by statistical means) against the data. In contrast, data mining is discovery driven in that useful patterns are automatically extracted from data. In order to accomplish this task, data mining systems frequently utilize methods from disciplines such as artificial intelligence, machine learning and pattern recognition.

The data mining algorithms discussed in this document operate on data sets composed of vectors (instances) of independent variables (or features). For example, a database may describe a group of automobiles in terms of their fuel type, number of doors, horsepower, engine location, length, width, price, engine size etc. In this case, fuel type is an example of feature and each instance corresponds to a distinct automobile. The general data mining process is shown in Figure 1; the sub-process depicted in this flow chart include:

Data Selection / Sampling – The sheer size of databases make it impractical to process them in their entirety. As a result, it is often necessary to winnow the data in some manner or randomly select of instances for processing

Cleaning / Preprocessing – During this phase, the selected data is prepared for processing by the data mining algorithm. This can involve translating the data into an acceptable format or replacing missing or illegitimate entries.

Transformation / Reduction – The purpose of this phase is to revise and / or redefine the feature set. In many cases all the features included in a given data set are not required for prediction. In either instance, it may be desirable to create new features to facilitate the mining process.

Data mining- this refers to the application of selected data mining method to the data

Evaluation Criteria - In this phase the output of the miming algorithm is evaluated against goodness criteria. This is typically done to reduce the volume of information produced to that which is most useful or relevant

Visualization – This is the task of massaging the output to facilitate manual analysis. Information that can be easily understood has the best chance of becoming usable knowledge.

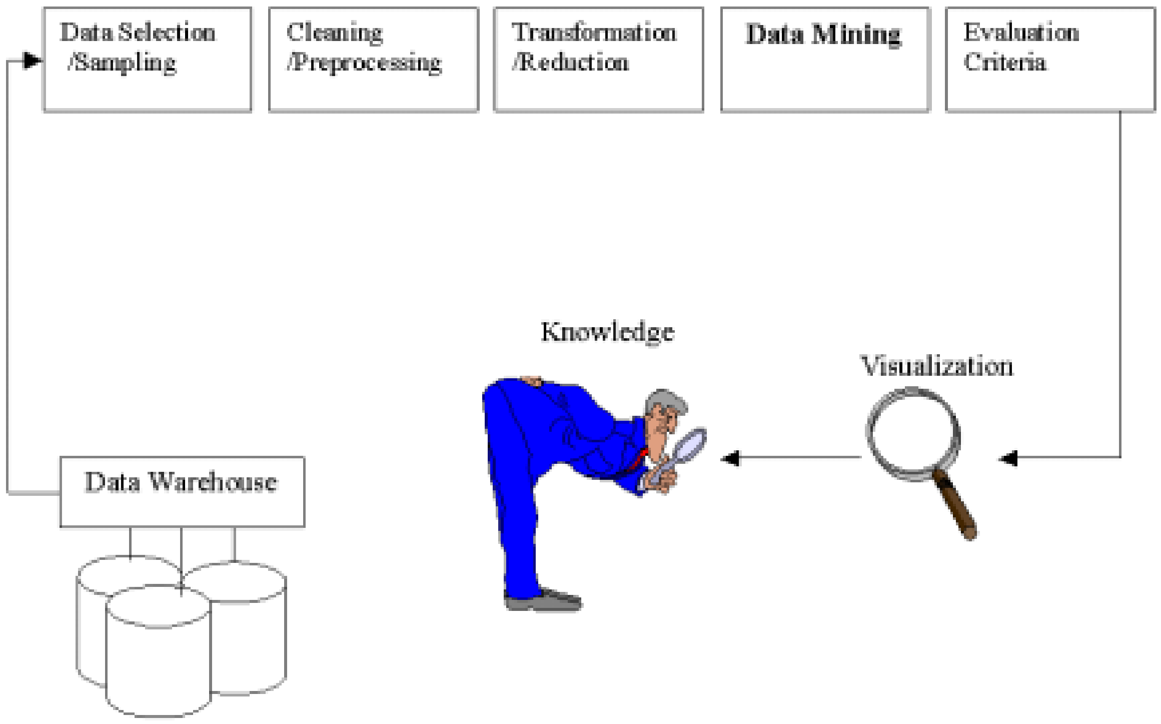


*Figure 1 Data mining process diagram*

### 1.1.2 Feature Subset Selection

The selection of appropriate features is an important precursor to most data mining methods. A good feature selection mechanism helps to facilitate classification by eliminating noisy or non-representative features that can impede recognition. Even features providing some useful information can reduce the accuracy of a classifier when the amount of training data is limited [2-4]. This so-called "*curse of dimensionality*", along with the expense of measuring and including features, demonstrates the utility of obtaining a minimum-sized set of features that allow a classifier to discern pattern classes well.

### 1.1.3 Feature Subset Weighting

Some classification rules, such as the K-nearest neighbors (KNN) rule, can be further enhanced by multiplying each feature by a weight value proportional to the ability of the feature to distinguish among pattern classes. Feature weighting is a variant of feature selection. It involves assigning a real-valued weight to each feature. The weight associated with a feature measures its relevance or significance in the classification task [5]. Feature subset selection is a special case of weighting with binary weights. This feature weighting method is a form of feature extraction defining new features in terms of the original feature set to facilitate more accurate pattern recognition. Feature selection and extraction, in combination with the K-nearest neighbors classification rule, have been shown to provide increased accuracy over the KNN rule alone. This method can aid in the analysis of large data sets by isolating combinations of features that distinguish “well” among different pattern classes [6,7].

Because each feature used as part of the classification procedure can increase the cost and the running time of a recognition system as well as reduce the accuracy of the result, there is strong motivation to design and implement systems using small feature sets. At the same time, there is a potentially opposing need to include a sufficient set of features to achieve high recognition rates under difficult situations.

Several authors have examined the use of the heuristic search for feature subset selection; this often operates in conjunction with a branch-and-bound search [8]. Others have explored randomized [9] and randomized, population-based heuristic search techniques such as genetic algorithms [10,11] to select feature subsets for use with decision-tree or nearest-neighbor classifiers.

Feature subset selection algorithms fall into two categories based on whether or not they perform feature selection independently of the learning algorithm that constructs the classifier. If the technique performs feature selection independently of the learning algorithm, it is defined as a “filter approach”. Otherwise, it follows a “wrapper approach” [8].

It is known that reducing the number of features increases the classification accuracy, but we assume that besides reducing the number of features, weighting features for the classification process may have more impact on the classification accuracy than feature selection

### 1.1.4 Algorithms For Feature Weighting and Feature Selection

GAs are search strategies based on the principles of natural selection. In a GA a population of possible solutions called chromosomes is maintained. Chromosomes are selected (each according to its fitness), recombined, and mutated to evolve a new population. The process is repeated until a “a stopping condition” has been achieved for the most-fit individual in the population or when a certain number of generations have been produced. GAs have been successfully used in a variety of optimization problems, and are best known for being robust search techniques that can search extremely large spaces and obtain globally competitive solutions.

## 1.2 Approach and Design Goals

This thesis effort investigates one data mining tool and the effectiveness and the efficiency of a developed GA algorithm that combines the feature weighting and feature selection methods in one technique in terms of classification accuracy and the execution time. The tool used is the Genetic Rule and Classifier Construction Environment (GRaCCE) developed by Marmelstein [19] and is discussed in Chapter 3. The primary emphasis is to analyze the GRaCCE and re-implement portions of it to increase the classification accuracy and to decrease the overall execution time.

The system under test is GRaCCE and the scope of this effort is limited to improvements of the classification accuracy via feature weighting and feature subset selection. Other improvements to overall GRaCCE algorithm such as using parallel processing techniques to reduce the overall execution time of the proposed serial algorithm are beyond the scope of this effort and are not researched

Thus, the investigation applies genetic algorithms to a feature subset weighting and feature subset selection problem. The goal is to evaluate the performance of the genetic algorithm for feature subset weighting and feature subset selection problem in terms of classification accuracy rate and to compare the classification accuracy of two different genetic algorithms when the encoding uses real-value strings for feature subset weighting and uses binary-value strings for feature subset selection.

## 1.3 Problem Scope

The portion of the feature subset weighting and feature subset selection problem addressed is the determination of the best features or attributes that can later be used to classify an unknown object, pattern, image or target. Reduction in the original feature size speeds up the learning process of any classifier and increase the classification accuracy of classifier system. The primary focus is increasing the classification accuracy rate by either feature subset weighting or feature subset selection encoding. Real-world datasets are used to train and validate the quality of our genetic algorithm. Then a very specific real-world dataset used to show that our genetic algorithm could solve a real world problem. No attempt is made to fine tune genetic variables (mutation rate, crossover rate, selection operator, and the others) for optimal performance. If tuning is to be done, a large number of various problems should be used to fine tune a genetic algorithm so that the fine tune applies to the general problem and not a specific problem. But many preliminary test experiments are conduced on the real-world datasets to find good values for the genetic algorithm parameters but these values are only decided for the feature subset weighting and feature subset selection problem and not for other optimization problems.

The reason behind the selection of this problem lies behind the desire to increase our classification accuracies on unknown objects and speed up the learning process. The new standardization and normalization algorithms, new genetic algorithm operators, new classification algorithms and new random number generators added to the GRaCCE environment provides better quality to this real-world application in solving the real-world problems.

## 1.4 Approach

The approach taken focuses on researching good genetic algorithm operators, feature subset weighting and feature subset selection encoding. Implementation makes extensive use of existing design, code, and test data. Six major objectives are addressed:

- Analyze the performance of genetic algorithm on real-world datasets

- Compare the performance of real-value genetic algorithm and binary-value genetic algorithm according to their classification accuracies on real-world datasets
- Compare the overall performance of the genetic algorithm with the other existing classification algorithms
- Propose genetic encoding of feature weight subsets and feature selection subsets
- Propose better genetic algorithm operators
- Integrate the new code to the existing GRaCCE environment

The problem domain is analyzed and the problematic areas are determined. The application is developed according to the user's requirements and expectations. The techniques to solve the problem is analyzed, their inefficiencies to solve the problem area are determined and these inefficiencies are optimized in the new algorithm.

## 1.5 Chapter Summary

In this chapter an introduction is given to data mining and the classification problem. The reason for feature weighting and feature selection is explained. The algorithms used for feature subset weighting and selection problem is introduced.

## 1.6 Outline of the Thesis

This thesis consists of six chapters. This chapter provides background information and defines the general problem of the thesis research. Chapter 2 focuses on historical perspective, problem domain models, and possible algorithm domains for solution of feature subset selection and feature weighting problem. Chapter 3 analyzes the problem, presents the possible high level; design solution, and narrows the focus of the research problem Chapter 4 presents the low-level design of the narrowed research problem. Chapter 5 gives the detailed justification of our methodology and experimental design process (parameter variations, hypothesis, number of tests, presentation techniques, experiment datasets). Chapter 6 includes the results of our experiments and Chapter 7 includes the statistical analysis of our experiment results. Chapter 8 presents the conclusions and the recommendations derived from the research effort.

# *II. Literature Review*

This chapter summarizes current literature related to data mining, feature subset weighting and feature subset selection, winnowing, k-nearest neighborhood classifier and genetic algorithms.

## 2.1 Data Mining

We live in the age of information where data is plentiful, to the extent that we are typically unable to process all of it usefully. Computer science has been challenged to discover approaches that can sort through the mountains of data available and discover the essential features needed to answer a specific question. These approaches must be able to process large quantities of data, in reasonable time and in the presence of "noisy" data i.e., irrelevant or erroneous data. One of these approaches is called "*data mining*" .

Data mining is the search for valuable information in large volumes of data. There are many other terms carrying a similar or slightly different meaning to data mining, such as *knowledge mining from databases, knowledge extraction, data/pattern analysis, data archaeology, and data dredging.* Many people treat data mining as a synonym for another popularly used term, "Knowledge Discovery in Databases", or KDD [1]. Alternatively, others view data mining as simply an essential step in the process of knowledge discovery in databases.

The real-world databases are highly susceptible to noise, outliers, missing, and inconsistent data due to their typically huge size, often several gigabytes or more.

## 2.2 Classification

Classification is an important aspect of the data-mining problem. A classification problem has an input data set called the training set, which consists of a number of examples each having a number of attributes. The attributes are either continuous, when the attribute values are ordered, or categorical, when the attribute values are unordered, or nominal when the attribute values are labeled. One of the categorical attributes is called the class label or the classifying attribute. The objective is to use the training data set to build a model of the class label based on the other attributes such that the model can be used to classify new data not from the training data set.

The discovered knowledge is usually represented in the form of IF-THEN prediction rules, which have the advantage of being a high-level, symbolic knowledge representation, contributing to the comprehensibility of the discovered knowledge. Aguilar and Riquelme put these decision rules in a decision queue (DQ), which means the obtained rules must be applied in specific order. With this policy, the number of rules may be reduced because the rules could be one inside of another. The user decides what representation provides the best accuracy and what produces the smaller number of rules. DQ presents the following structure:

```
If conditions Then class
Else            if conditions Then class
                Else            if conditions then class
                                ……………………………………
                                                Else “unknown class” [38]
```

The discovered rules could be evaluated according to several criteria, such as the degree of confidence in the prediction, classification accuracy rate on unknown-class examples, comprehensibility, cost, etc

Because each feature used as part of the classification procedure can increase the cost and running time of a classifier system as well reduce the accuracy of the result, there is a strong motivation to design and implement systems using small feature sets. Feature weighting is a variant of feature selection. It involves assigning a real-valued weight to each feature. The weight associated with a feature measures its relevance or significance in the classification task. Feature subset selection is a special case of weighting with binary weights.

## 2.3 Feature Subset Selection

The feature subset selection problem is well known in statistics and patterns recognition. However, many of the techniques deal exclusively with features that are continuous, or, make assumptions that do not hold for many practical machine-learning algorithms. For example, one common assumption (monotonicity) says that increasing the number of features can never decrease the performance.

Although assumptions such as monotonicity are often invalid for machine learning, one approach to feature subset selection in machine learning has borrowed search and evaluation techniques from statistics and patterns recognition. This approach dubbed the wrapper approach and estimates the accuracy of the feature subsets via a statistical re-sampling technique (such as cross validation) using the actual machine-learning algorithm. The approach has proved to be useful but is very slow to execute as the induction algorithm is called respectively.

Another approach to feature subset selection, called filtering, operates independently of any induction algorithm- undesirable features are filtered out of the data before induction begins. Filter methods typically make use of all the training data when selecting a subset of features. Some looks for consistency in the data. That is, they note when every combination of values for a feature subset is associated with a single class label. Another eliminates the features whose information content (concerning other features and the class) is subsumed by some number of the remaining features. Still other methods attempt to rank features according to a relevancy score. Filters have proven to be much faster than wrappers and hence can be applied to large data sets containing many features.

Computational studies of Darwinian evolution and natural selection have led to numerous models for solving optimization [12 - 16]. Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They have been developed by John Holland and have drawn increasing interest. GA’s comprise a subset of these evolution-based optimization problems techniques focusing on the application of selection, mutation, and recombination to a population of competing problem solutions [17], [18]. GA’s are parallel, iterative search algorithms, and have successfully applied to a broad spectrum of optimization problems, including many pattern recognition and classification tasks. In our work, a genetic algorithm is designed to improve feature weights used in the distance calculation of KNN.

The problem of dimensionality reduction is well suited to formulation as an optimization problem. Given a set of d-dimensional input patterns, the task of the GA is to find a transformed set of patterns, in an m-dimensional space (m<d) that maximizes a set of optimization criteria. Typically, the transformed patterns are evaluated based upon both their dimensionality, and their class separation or the classification accuracy. The GA maintains a population of computing feature transformation matrices. To evaluate each matrix in this population, the input the matrix, producing a set of transformed patterns, which are then sent to a classifier, multiplies patterns. The available samples are divided into a training set, used to train the classifier, and a testing set, used to evaluate classification accuracy. The accuracy obtained is then returned to the GA as a measure of the quality of the transformation matrix used to obtain the set of transformed patterns. Using this information, the GA searches for a transformation that minimizes the dimensionality of the transformed patterns while maximizing the classification accuracy.

A direct approach to using GA's for feature selection was introduced by Siedlecki and Sklansky [19]. In their work, a GA is used to find a good binary vector, where each bit is associated with a feature. If the *i*th bit of this vector equals 1, then the *i*th vector is allowed to participate in classification; if the bit is 0, then the corresponding feature does not participate (see Figure 2). Each resulting subset of features is evaluated according to its classification accuracy on a set of testing data using K-nearest neighbor classifier.

The single bit associated with each feature is expanded to a real-valued coefficient, allowing independent linear scaling of each feature, while maintaining the ability to remove features from consideration by assigning a weight of zero. Given a set of feature vectors of the form $X= \{x_1, x_2, x_3,…., x_d\}$, the GA produces a transformed set of vectors of the form $X^{'}= \{ w_1x_1, w_2x_2, w_3x_3,…., w_dx_d \}$ where $w_i$ is a weight associated with feature *i*. Each feature value is first normalized, and then scaled by the associated weight prior to training, testing, and classification. This linear scaling of features prior to classification allows a classifier to discriminate more finely along feature axes with larger scale factors. A

KNN classifier is used to evaluate each set of feature weights. The effects of linear feature weighting on the KNN classification rule are visualized in Figure 3. Patterns plotted in feature space are spread out along feature axes with higher weight values. The value of K for KNN classifier is fixed and determined empirically prior to feature extraction.

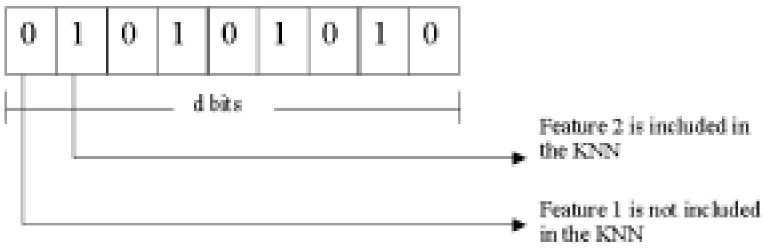

0 1 0 1 0 1 0 1 0
d bits
Feature 2 is included in the KNN
Feature 1 is not included in the KNN

*Figure 2 d-dimensional binary feature vector sGA-based feature selection*

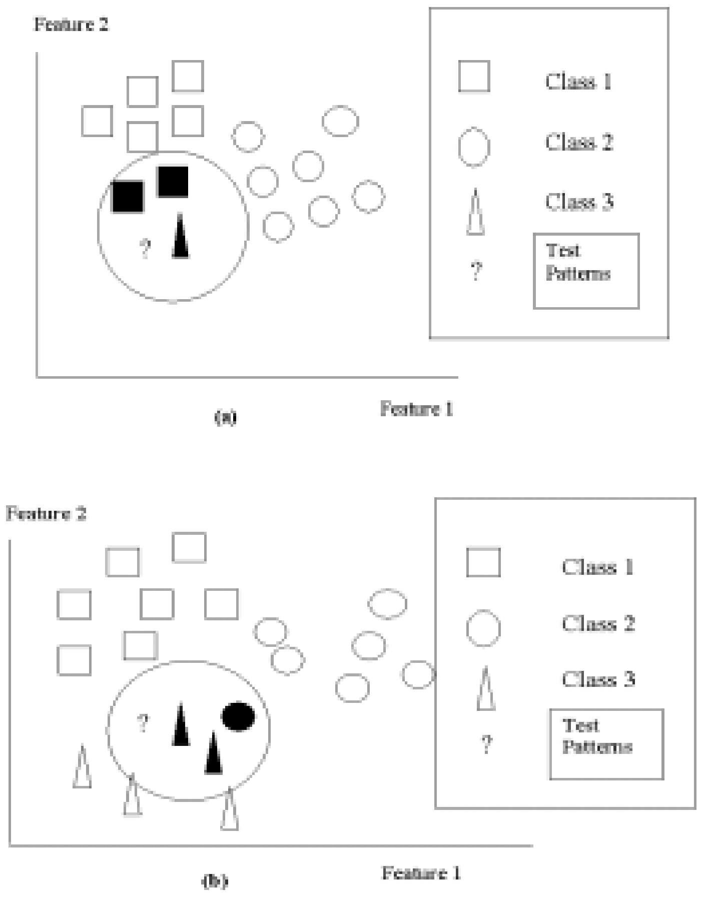


*Figure 3 (a) Original data. (b) Scaled data.*

## 2.4 Winnowing

Class overlap within a given data set can impede the rule induction process and degrade the quality of the generated rule set. Since the final product of a data mining process is the rules generated for a specific problem domain, a data mining application must be able to generate these rules, which will be presented to the decision makers. Feature subset selection and weighting does not clear these class overlaps but feature subset selection and feature weighting process finds the features which have better classification accuracy and causes less class overlaps when compared to original feature sets. Since these class overlaps cannot be removed by feature selection and feature weighting technique, a technique, which is called "winnowing", is used to get rid of these class overlaps. Since these class overlaps degrade the quality of the data mining rules or prevent extracting these rules from the system, it is therefore desirable to make the classes piecewise, linearly separable. Wagner's edited nearest neighbor technique [25] provides an elegant way of accomplishing this goal. Wagner's method removes data points from the training set that are misclassified by a KNN algorithm but this time instead of using original feature set, reduced feature set can be used to find misclassified points. This process continues until it converges to a subset of the data with no classification error. Devijer and Kittler [26] proved that this procedure is asymptotically Bayes-optimal with respect to the original data set. Winnowing can be performed on either the full feature set or on a specified feature subset. As an added benefit, there is evidence showing that this procedure contributes to a reduction in rule set size (for further reference Marmelstein 1999, [19]).

## 2.5 K-Nearest Neighbor Algorithm

This classifier classifies a pattern x by assigning it to the class label that is most frequently represented among it's k nearest pattern. In the case of a tie, the test pattern is assigned the class with minimum average distance to it. Hence, this method is sensitive to the distance function. For the minimum average distance, the metric employed is the Euclidean distance. This metric requires normalization of all features into the same range. The k-nearest neighbor classifier is a conventional non-parametric classifier that is said to yield good performance for optimal values of k.

In KNN classification, each training pattern is represented in a d-dimensional space according to the value of each its d features. The test pattern is then represented in the same space, and its k nearest neighbors, for some constant k, is selected. Neighbors are usually calculated according to Euclidean distance, although other metrics (e.g., Mahalanobis distance) are sometimes used. The class of each of these k neighbors is then tallied, and the class with the most "votes" is selected as the classification of the test pattern. The simplicity of the KNN classifier makes it relatively easy to implement. Its non-parametric nature allows classification of a broad range of data sets, including those for which feature values do not follow a multivariate Gaussian distribution. Due to the use of Euclidean distance metric, the KNN decision rule is sensitive to scaling of the feature values- a necessary prerequisite for use with a weighted-GA. Classifiers based on the class-conditional distributions of the feature values, such as Bayes classifier, are invariant to scaling of the feature values. Finally, the KNN classifier is well explored in the literature, and has been demonstrated to have good classification performance on a wide range of real-world data sets. The asymptotic error rate of KNN decision rule can be shown to be bounded by the Bayes error as k $\rightarrow$ infinity [20].

## 2.6 Data Standardization and Normalization

An attribute is normalized by scaling its values so that they fall within a small-specified range, such as 0 to 1.0. Normalization is particularly useful for classification algorithms involving neural networks, or distance measurements such as nearest-neighbor classification and clustering. If using the neural network back propagation algorithm for classification mining, normalizing the input values for each attribute measured in the training samples helps speed up the learning phase. For distance-based methods, normalization helps prevent attributes with initially large ranges from outweighing attributes with initially smaller ranges. There are many methods for data normalization such as min-max normalization, z-score normalization, and normalization by decimal scaling.

Min-max normalization performs a linear transformation on the original data. Suppose that $\min_A$ and $\max_A$ are the minimum and maximum values of an attribute A. Min-max normalization maps a value v of A to v′ in the range [new_ $\min_A$ , new_ $\max_A$] by computing

$$v' = v - \min_A / \max_A - \min_A \times (\text{new\_} \max_A - \text{new\_} \min_A) + \text{new\_} \min_A. \qquad (2.1)$$

Min-max normalization preserves the relationship among the original data values. It encounters an "out of bounds" error if a future input case for normalization falls outside of the original data range for A.

In z-score normalization (or zero-mean normalization), the values for an attribute A are normalized based on the mean and standard deviation of A. A value v of A is normalized to v' by computing

$$v' = v - A(\mu) / \sigma_A \qquad (2.2)$$

Where $A(\mu)$ and $\sigma_A$ are the mean and standard deviation, respectively, of attribute A. This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliners which dominate the min-max normalization.

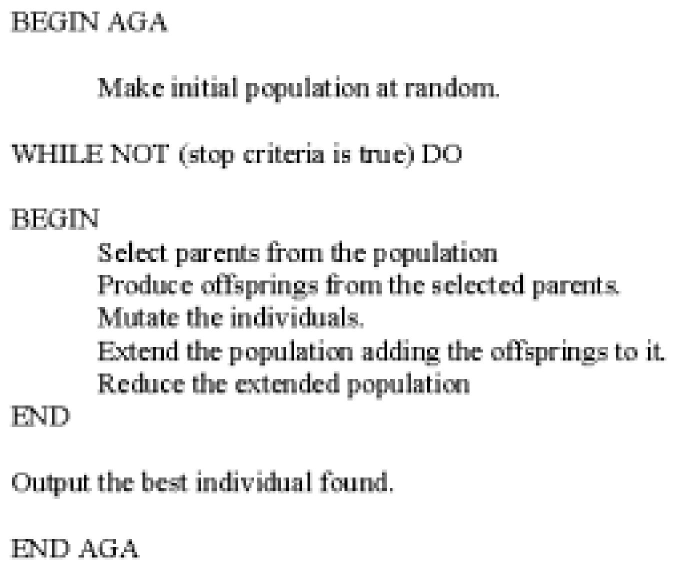
Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A. A value v of A is normalized to v' by computing

$$v' = v / 10^{j'} \qquad (2.3)$$

Where j is the smallest integer such that $\text{Max}(|v'|) < 1$.

## 2.7 Genetic Algorithms

Holland (1975) introduced genetic algorithms. In these algorithms the search space is represented as a collection of individuals. The individuals represent candidate solutions to the optimization problem being solved. The purpose of using genetic algorithm is to find the individual from the search space with the best "genetic material" . A wide range of genetic representations (e.g., bit vectors, LISP programs, matrices, etc.) can be used to encode the individuals depending on the space of solutions that needs to be searched. In genetic algorithms, the individuals are represented by either integer-value strings (or matrices) or real-value (or matrices), which are often called chromosomes. The resulting search space corresponds to an n-dimensional Boolean space. In the feature subset selection problem, each individual would represent a feature subset. The quality of an individual (or fitness of the individual in the population) is measured with an evaluation function. In the feature subset selection problem, the fitness function would evaluate the selected features with respect to some criteria of interest (e.g., cost of the resulting classifier, classification accuracy of the classifier, etc.). The part of the search space to be examined is called the population. Roughly, a genetic algorithm works as follows (see Figure 4).



```
BEGIN AGA

        Make initial population at random.

WHILE NOT (stop criteria is true) DO

BEGIN
        Select parents from the population
        Produce offsprings from the selected parents.
        Mutate the individuals.
        Extend the population adding the offsprings to it.
        Reduce the extended population
END

Output the best individual found.

END AGA
```

*Figure 4 The pseudo-code of the Abstract Genetic Algorithm (AGA)*

First, the initial population is chosen, and the quality of this population is determined, next, in every iteration parents are selected from the population. These parents produce children (or offspring), which are added to the population. For all newly created individuals of the resulting population a probability near to zero exits that they will “mutate” , i.e. they will change their hereditary distinctions.

After that, some individuals are removed from the population according to a selection criterion in order to reduce the population to its initial size. One iteration of the algorithm is referred to as a generation.

### 2.7.1 Selection Operator in GAs

Evolutionary algorithms use some form of fitness-dependent probabilistic selection of individuals from the current population to produce individuals for the next generation. A variety of selection techniques have been explored in the literature. Some of the most common ones are *fitness-proportionate* selection, *rank-based* selection, and *tournament-based* selection .he selected individuals are subjected to the action of genetic operators to obtain new individuals that constitute the next generation. The operators, which define the child production process and the mutation process, are called crossover and mutation operator respectively. The genetic operators are usually designed to exploit the known properties of genetic representation, the search space, and the optimization problem to be solved. Mutation and crossover play different roles in the genetic algorithm. Mutation is needed to explore new states and helps the algorithm to avoid local optima. Crossover should increase the average quality of the population. By choosing adequate crossover and mutation operators, the probability that the genetic algorithm results in a near-optimal solution in reasonable number of iterations is increased. Genetic operators enable the algorithm to explore the space of candidate solutions. Crossover and Mutation

Mutation operates on a single string and generally changes a bit at random. Thus, a string 11010 may, as a consequence of random mutation, get changed to 11110. Crossover, on the other hand, operates on two parent strings to produce two offspring. With a randomly chosen crossover position 4, two strings 01101 and 11000 yield the offspring 01100 and 11001 as a result of crossover. Other genetic representations (e.g., matrices, LISP programs) require the use of appropriately designed genetic operators There are many different crossover and mutation operators and these operators are summarized in Table 1.

*Table 1 Summary of representations and operators*

| Representation | Operators | Authors |
|---|---|---|
| Binary | Classical + repair operator | Lidd (1991) |
| Path | Partially- Mapped Crossover | Goldberg and Lingle (1985) |
| | Order Based Crossover | Syswerda (1991) |
| | Position Based Crossover | Syswerda (1991) |
| | Heuristic Crossover | Grefenstette (1987b) |
| | Edge Recombination Crossover | Whitley et al. |
| | Sorted Match Crossover | Brady (1985) |
| | Maximal Preservative Crossover | Muhlenbein et al. (1988) |
| | Voting Recombination Crossover | Muhlenbein (1989) |
| | Alternating- Position Crossover | Larranaga et al. (1996a) |
| | Displacement Mutation | Michalewicz(1992) |
| | Exchange Mutation | Banzaf (1990) |
| | Insertion Mutation | Fogel (1990) |
| | Simple Inversion Mutation | Holland(1975) |
| | Inversion Mutation | Fogel (1990) |
| | Scrmable Mutation | Syswerda (1991) |
| Adjacency | Alternating Edge Crossover | Grefenstette et al. (1985) |
| | Subtour Chunks Crossover | Grefensette et al. (1985) |
| | Heuristic Crossover 1 | Grefensette et al. (1985) |
| | Heuristic Crossover 2 | Jog et al. (1989) |
| | Heuristic Crossover 3 | Suh and Van Gucht (1987) |
| Ordinal | Classical operators | Grefenstette et al. (1985) |
| Matrix | Intersection Crossover Operator | Fox and Mc Mahon (1987) |
| | Union Crossover Op [erator | Fox and Mc Mahon (1987) |

As the primary sources of exploration, crossover and mutation play important roles in GA search process. Many of these roles relate to building blocks: segments of an individual that affect that individual’s fitness.

Crossover is expected to propagate existing building blocks to the next generation. By manipulating “chunks” of an individual rather than individual bits, crossover is expected to be able to transfer entire building blocks from parent to offspring. Crossover is able to make potentially large changes from parents to offspring with minimal distribution of building blocks [28].

Crossover is expected to combine existing building blocks. Lower order building blocks are more likely to be found through random sampling and can be searched for in parallel on different members of the GA population. Higher order building blocks are easier to construct from combining lower order building blocks than from random sampling [28].

Both crossover and mutation are expected to construct new building blocks and disrupt existing building blocks. [28]

Mutation is expected to introduce diversity. Since crossing over identical segments provides no new information, once the entire population possesses the same value at a particular bit position, only mutation is able to insert new values of that bit into the population. [28]

As the population becomes more homogeneous, crossover becomes less productive [28]. Mutation provides some of the necessary diversity to a GA population; it is considerably more active than crossover at destroying building blocks and as good or better than crossover at construction new building blocks. Increasing the genome length, and consequently the amount of non-coding material makes crossover less active within building block boundaries, but does not affect mutation's constructive and destructive effects on building blocks.

### 2.7.2 Infeasible Solutions and Repair Algorithms

Repair algorithms enjoy a particular popularity in the evolutionary computation community: for many combinatorial optimization problems (e.g. the traveling salesman problem, the knapsack problem, and the set covering problem) it is relatively easy to repair an infeasible individual. Such repaired version can be used either for evaluation only; that is, $eval_u(y) = eval_f(x)$

Where x is a repaired (i.e. feasible) version of y, or it can also replace (with some probability) the original individual in the population. [29]

The crossover and mutation operators may not always produce feasible individuals, but the individuals that are evaluated by fitness function must be feasible individuals. To prevent infeasible individuals from taking place in the population, *repair algorithms* are developed. Repair algorithms are user-specified algorithms and are selected according to the criteria of feasibility for the problem domain. For the feature weighting problem, the sum of weights has to be equal to 1, on the other hand the since the crossover operator produce new individuals by changing the genes of parent chromosomes randomly, the new individuals may not necessarily be feasible (the feasible individuals have a sum of 1). If real-value chromosomes are used as in feature weighting problem, ordinary crossover operators (e.g., 1-point crossover, 2-point crossover, Uniform crossover) always produce infeasible solutions, and to prevent these infeasible individuals, either a repair algorithm has to be defined or a new recombination operator has to be defined. For the later solution, Demiroz and Guvenir proposed a new crossover operator, which is called continuous uniform crossover (CUCO) operator, that guarantees the legality of the offspring. Given two chromosomes $x=<x_1, x_2,.....,x_n>$ and $y=<y_1, y_2,......,y_n>$ such that n is the number of genes in a chromosome, the offspring are defined as $x'=<x_1',x_2',.....,x_n'>$ and $y'=<y_1', y_2',....,y_n'>$, where

$$x_i'=s \times x_i + (1-s) \times y_i$$

$$y_i'=s \times y_i + (1-s) \times x_i$$

Here s, called stride, is constant through a single crossover operation. This crossover operation guarantees that the sum of the alleles of an offspring is still 1 given that the sum of the parents' alleles is 1.

The process of fitness-dependent selection and application of genetic operators to generate successive generations of individuals is repeated many times until a satisfactory solution is found (or the search fails). It can be shown that evolutionary algorithms of the sort outlined above simulate highly opportunistic and exploitative randomized search that explores high-dimensional search spaces rather effectively under certain conditions In practice, the performance of evolutionary algorithms depends on a number of factors including: the choice of genetic representation and operators, the fitness function, the details of the fitness-dependent selection procedure, and the various user-determined parameters such as population size, number of generations, probability of application of different genetic operators, etc.

There can be various criteria for stopping GA. For example, if it is possible to determine the number of generations required for a good solution before hand. But the stopping criteria should normally take into account the uniformity of the population, the relationship between the average objective function with respect to the objective function of the best individual, as well as not producing an increase in the objective function of the best individual during a fixed number of cycles. One possible stopping criteria for feature subset selection and weighting problem is, if the classification accuracy keep decreasing on the test set for a while the classification accuracy increases on the training set, a threshold value can be defined and when this threshold value is reached, the execution of GA can be stopped by assuming that because of the bias on the classifier to the some features because of the characteristics of the train set, the solution that is about to be found can not be generalized to test set and to the future unknown patterns. An important aspect of Gas is that, GAs can return the best, complete solution at any given time. This characteristic is important especially for the problems where time is important and solution is needed in a short time. Further description of genetic algorithms can be found in Goldberg (1989) and Davis (1991).

## 2.8 Chapter Summary

In this chapter, we explained the problem domain in detail. The data mining process is introduced and the literature about the feature subset weighting and selection problem is reviewed. KNN Classifier explained and different normalization techniques to normalize the original feature values in any data set are introduced. The idea of winnowing is explained. We also introduced the necessary information to understand the sGA Algorithm, the genetic operators and repair functions to repair the infeasible solutions.

# *III. Design Methodology*

## 3.1 Introduction

This chapter discusses the design methodology we followed to develop our solution for the feature subset weighting and selection problem

## 3.2 Genetic Rule and Classifier Construction Environment

GRaCCE is a data-mining tool that uses evolutionary search techniques [24] to mine classification rules from the data given. It is similar to a pattern recognition algorithm, but goes beyond by producing understandable rules used in the recognition. GRaCCE is designed for use with samples in a vector form [25]. The individual elements in the vector are independent features, which may be real-values or discrete. Getting data into this form is the primary goal of the first step in the data mining process.

Once the data is in the proper form, GRaCCE employs a pre-processing phase that includes an optional feature selection operation and/ or a mandatory winnowing operation. Winnowing removes statistical outliers and duplicate samples resulting in classes of data that are linearly separable. GRaCCE then selects the relevant data sets for training through either a random sampling of data or a cross-validation method. Next, it executes a data-mining algorithm that can be viewed as a combination of clustering and classification, since it forms a classification model of data based on data attributes that include the clustering of class homogeneous regions. The steps that the data-mining algorithm follows sequentially are: partition generation, data set approximation, region identification, region refinement, and partition simplification. Worst case analysis by section is shown in Table 2, where n is the number of samples and d is the number of features. The results of the data-mining algorithm are finally interpreted and evaluated through a listing of partitions that form these regions in the data.

| MODULE | COMPLEXITY |
|---|---|
| Pre-processing – Feature Weighting | $O(n^2 + kn)$ |
| Pre-processing – Feature Selection | $O(n^2 + kn)$ |
| Pre-processing - Winnowing | $O(dn^2d)$ |
| Partition Generation | $O(2(dn)^2 + nd)$ |
| Data Set Approximation | $O(nlog(n))$ |
| Region Identification | $O(qn^2d^{3/2})$ |
| Region Refinement | $O(dnlog(n))$ |
| Partition Simplification | $O(d^2n^2log(n))$ |

*Table 2 Worst Case Time Complexity Analysis of GRaCCE*

At the top level is the root GRaCCE process gracce. GRaCCE has three main modules at the second level along with the load module. The load module is used to read in the data set to train the KNN classifier prior to any of the other modules being called. The first of four main modules is the feature-weighting module. The other three modules are feature selection, winnow and ruleInd. The rule induction module makes the most extensive use of the rest of the GRaCCE code and calls the five other modules. The ga module provides method for both the feature selection routines along with the region ID section of the rule induction phase [34]. For further information about GRaCCE see Appendix A.

### 3.2.1 Objective vs. Fitness Structure

Most evolutionary computation (EC) algorithms require some form of encoding or mapping [26] between the problem domain and the algorithm domain. The evolutionary operators operate on these encoded structures sometimes called chromosomes. GRaCCE uses either binary or real value string representation for each population member with a length determined by the number of features in the data set. In the feature selection process each feature is represented by a single binary digit. On the other hand for feature-weighting process each weight is associated with each feature is represented by a real value digit. Feature selection can be thought as a special implementation of feature weighting where weights take only the values of 0 and 1. For feature selection process a one represents the feature is present and a zero represents the feature is absent. For feature weighting process a value other than zero represents the feature is present and has this value as its weight in labeling the associated class, and a zero represents as in feature selection process that the feature is absent.

The fitness landscape [27] for an EC algorithm refers to the result of the mapping between the genomes of a population of individuals to their fitness and a graphical representation of that mapping. For GRaCCE, each population member is evaluated using the objective function in either the feature selection or the feature weighting code. The objective function performs a k-nearest neighbor (kNN) routine over a subset of data for each member on the population. The subset is determined by the bit representation of the individual member.

GRaCCE's fitness function gets the results of the objective function and finds the k-nearest neighbors of the unknown instance (each gene in the chromosome represents a feature) according to the calculated distances on each feature dimension. The new algorithm does not sort the calculated distances to find the smallest k distance values and their associated classes. Since k is usually a very small integer number, instead of sorting (sorting is expensive and inefficient since if there are n examples, n-1 distance values have to be sorted and since n is often a large integer number it is a huge bottleneck when this sort operation is called many times), the distance matrix is searched for the first smallest distance value and when found this distance value and its associated class label is read and the class number in the class matrix is incremented and then the distance matrix is searched for the second smallest distance value, but before starting the second search the first minimum distance value is changed to infinity to prevent finding the same distance value again. This operation continued for k times until the smallest k distance value is found and their associated class labels are read. The class, which has the majority in the class matrix, is selected as the class label of the unknown instance. Then the confusion matrix is built to evaluate the performance of any chromosome (feature vector). The values on the diagonal of the confusion matrix shows the number of correctly classified classes and the values out of diagonal of the matrix shows the number of misclassified classes by the chromosome that is being evaluated. The fitness value of a chromosome is calculated after this step and the percentages of misclassified classes to total number of classes in the system gives the classification error rate of any chromosome and this value is the fitness value of a feature subset that is being evaluated. The objective score for a particular member does not change for the entire execution of the genetic algorithm (GA) as long as its binary string remains unchanged. However, the fitness of that member is recalculated and may change with each generation. The fitness landscapes for data Fisher's Iris dataset (Iris dataset is explained in Chapter 5 in detail) under original and noise added conditions to these data sets are shown in Figure 7, Figure 8, Figure 9 and Figure 10. Please see Appendix A. for the landscapes of other real-world data sets.

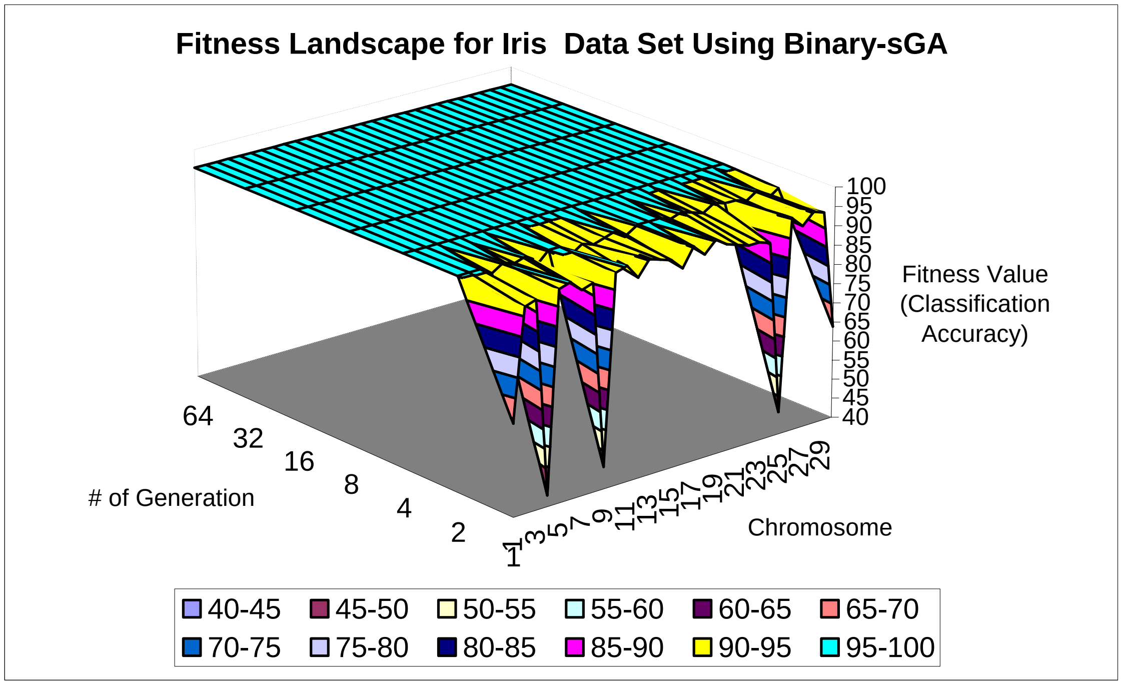


*Figure 5 Fitness Landscape for Iris Data Set Using Binary-sGA*

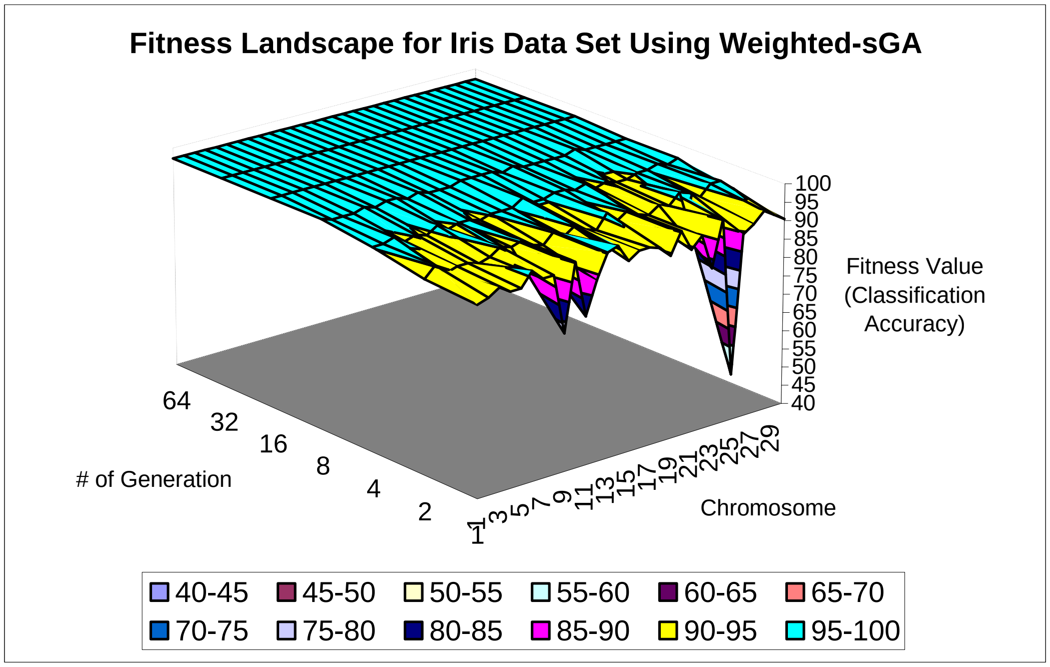


*Figure 6 Fitness Landscape for Iris Data Set Using Weighted-sGA*

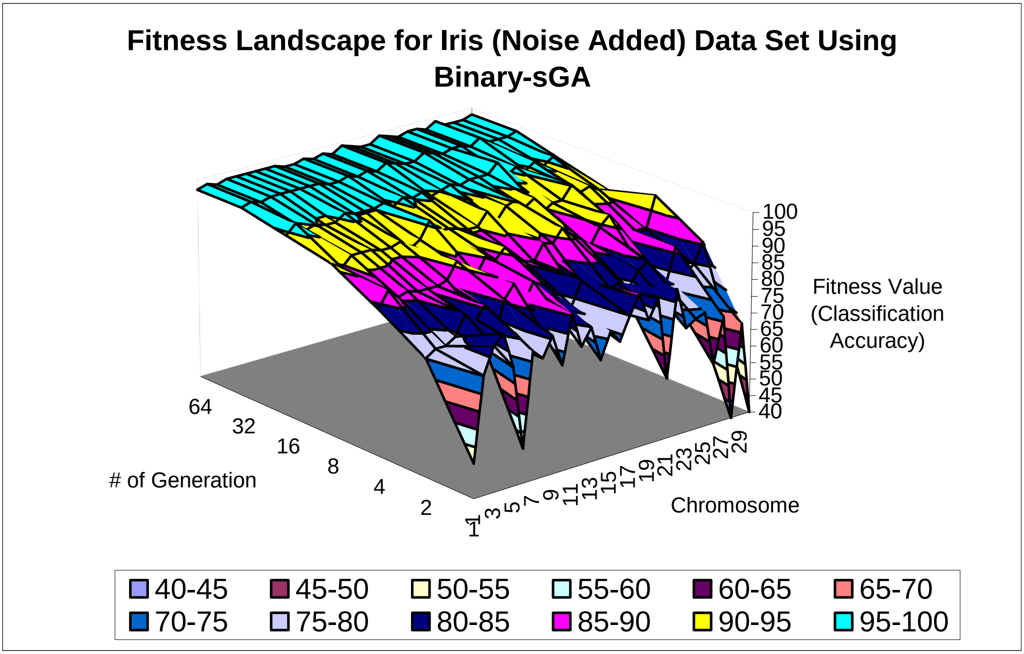


*Figure 7 Fitness Landscape for Iris (Noise Added) Data Set Using Binary-sGA*

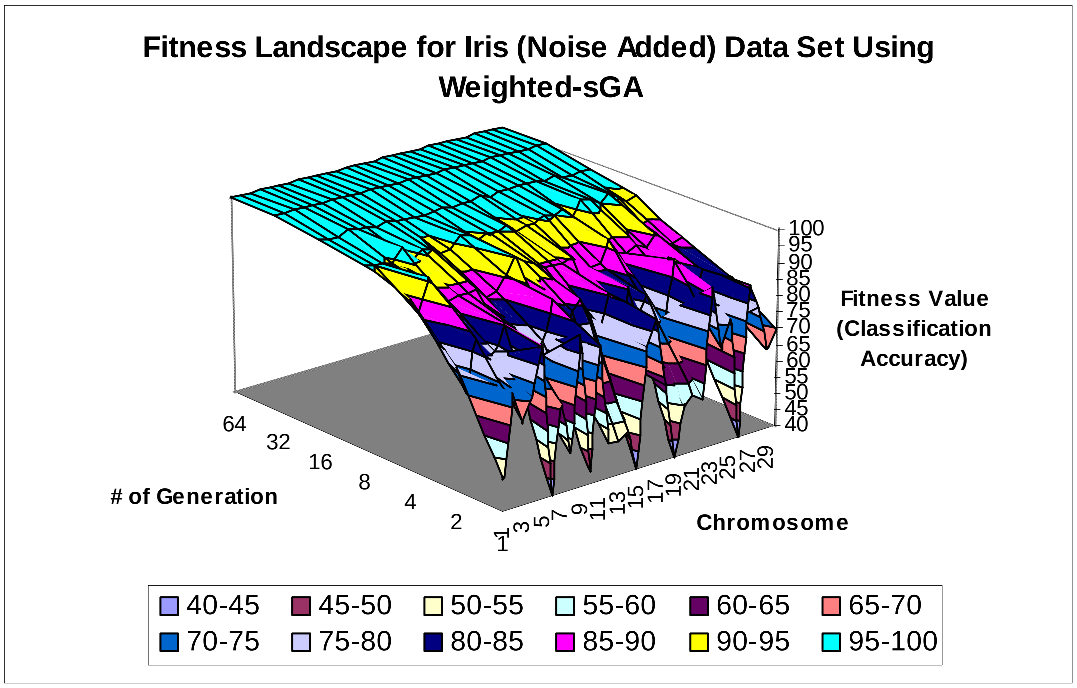


*Figure 8 Fitness Landscape for Iris (Noise Added) Data Set Using Weighted-sGA*

## 3.3 Chapter Summary

In this chapter we explained our design methodology and introduced detailed information about the Genetic Rule and Classifier Construction Environmnet (GRaCCE). We also introduced the objective and fitness landscape of our sGA algorithm for weighted classification and binary classification.

# *IV. Implementation*

## 4.1 Introduction

This chapter discusses the implementation of chromosome encoding, genetic algorithm operators, proposed the repair algorithms and proposed objective and fitness functions.

## 4.2 GRaCCE Environment

GRaCCE is developed under Microsoft Visual C++ Version 6.0. GRaCCE mainly uses dynamic memory allocation and for that purpose TNT library is integrated into GRaCCE. This library controls every dynamic memory allocation. TNT library is developed according to object-oriented method. When the dynamic memory array object is created, the object constructor is called and if requested memory is allocated. The memory is not allocated to an array object until the memory is requested by the command:

```
Object = Object.newsize(row_size, col_size)
```

The memory is deallocated automatically when the object goes out of scope. For example, if the object created in a function, the memory is automatically deallocated for this array object automatically when the operation returns to the calling function.

Since Visual C++ does not support Garbage Collection inherently as Java supports, using TNT Matrix Library overcomes this disadvantage of Visual C++.

The initial population for either weighted-sGA or binary-sGA is initialized by using Visual C++ Version 6.0 random number generator, and then for other operations such as selection, crossover and mutation, a uniformly distributed random generator used as explained in Appendix C.

## 4.3 Loading the Data Set

To be able to load the original datasets into GRaCCE environment, the data sets has to be formatted into correct format that is suitable for the GRaCCE. Since GRaCCE accepts only numeric values, if there are any non-numeric values in the original datasets, they have to be formatted into correct format. GRaCCE assumes that in the first column of the each dataset there is the class label for each instance and the remaining columns are the feature columns. Each row in the datasets describes an instance. When the dataset is loaded into GRaCCE environment, the original dataset is read row by row by using Microsoft Visual C++ command “getline”. This command ensures that each row is read as a whole, and then each row is stored into an array of linked list queue. After the reading of the original dataset is finished and all data in the original dataset is stored into an array of linked list queue, the array of linked list is transformed into an globally defined dataset matrix. This matrix remains constant until the end of the program execution is stopped.

After the data set is loaded into GRaCCE environment by load module, the original dataset is first normalized. There are three different normalization (explained in Chapter 2) method for this purpose as shown in Table 3.

| |
|---|
| Min-max normalization |
| z-score normalization |
| Normalization by decimal scaling |

*Table 3 Normalization Techniques used in GRaCCE*

Normalization of the original dataset is followed by the division of the normalized original dataset into training set and test set. The normalized original dataset is divided into training and test set according to user specified division rates and usually the rate for training set is given greater than rate for the test set.

## 4.4 Encoding Scheme

Although the actual encoding scheme is same for both weighted-sGAS and binary s-GA, the goal of the encoding for two algorithms is different. The encoding of the chromosomes for binary-sGA is selected as a way that the each gene in the chromosome shows if feature is selected or not, and the encoding of the chromosome for weighted-sGA is selected as a way that each gene in the chromosome shows if the feature is important or not. If the feature is important it has the highest weight and if the feature is not important the feature has zero weight end eventually is not selected. The length of chromosomes in both algorithms is equal to the number of original features in the problem dataset. Figure 11 shows the proper encodings for two algorithms.

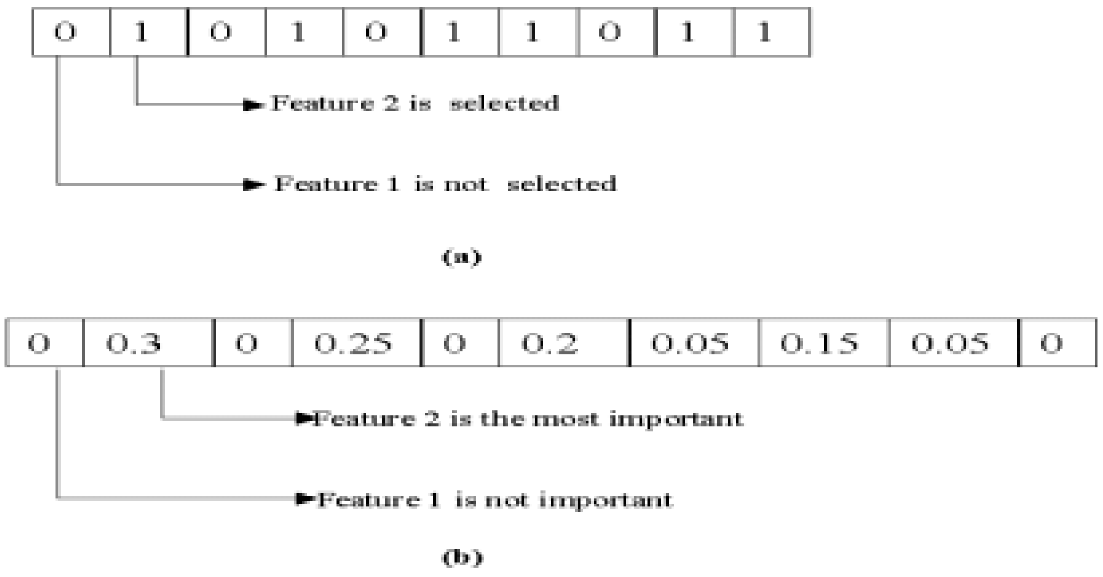

0 1 0 1 0 1 1 0 1 1
Feature 2 is selected
Feature 1 is not selected
(a)
0 0.3 0 0.25 0 0.2 0.05 0.15 0.05 0
Feature 2 is the most important
Feature 1 is not important
(b)

*Figure 9 (a) Encoding in a binary-SGA (b) Encoding in a weighted-sGA*

The binary encoding of the chromosome in binary-sGA makes it suitable for the original sGA operators. For example the crossover operators such as one-point crossover and two-point crossover can easily be implemented for binary encoding and the inversion mutation also can be easily implemented on a binary chromosome. This is the most important advantage of the binary-value chromosome over a real-value chromosome. The crossover and mutation operators are easier and simpler in binary encoding. The other advantage of binary encoding is that the crossover and mutation operators don’t produce any infeasible chromosomes while searching for the solution of the feature subset selection problem. But having these advantages don’t necessarily mean that binary encoding could be used to search the solution to every problem.

The real-value encoding is used for weighted-sGA and it is necessary that this encoding does not produce any infeasible chromosomes during the search of the solution. For the feature weighting and selection problem a chromosome is said to be feasible if the sum of all weights is equal to 1 and this is shown below:

$$W = \sum_{i=1}^{n} w_i = 1 \tag{4.1}$$

Where $w_i$ represents the ith gene and it's value in the chromosome and W represents the sum of all weights in a chromosome.

But as long as crossover and mutation operators are used, infeasible chromosomes are inevitable during the search process. The solution to this problem is using repair algorithms as explained in Chapter 2. Figure 12 shows infeasible offspring chromosomes produced by one-point crossover of two parents. Before the one-point crossover operation the chromosomes are feasible but after the one-point crossover operation the new chromosomes are infeasible and the fitness values calculated for these chromosomes does not represent a correct result. The solution to this problem is sending these chromosomes to a repair algorithm and changing then to feasible chromosomes.

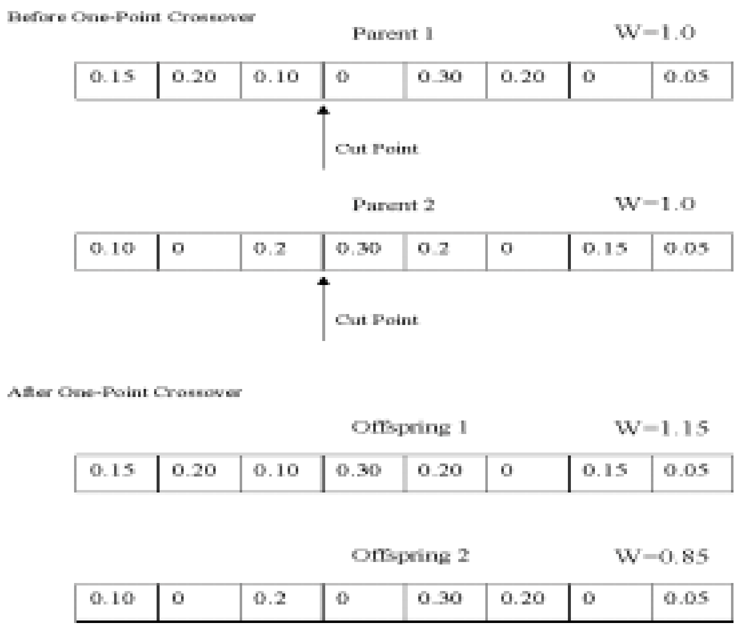


*Figure 10 Infeasible children after one-point crossover*

## 4.5 The Repair Algorithm

The repair algorithm we use is shown below. The algorithm adds each gene value to calculate the total sum weight for any chromosome. The algorithm then divides each gene with this sum. The algorithm guarantees that $0< w_i \leq 1$ and W = 1. Figure 13 shows the feasible chromosomes for the previous example.

**Input:** Idata, input row vector or chromosome, which is infeasible

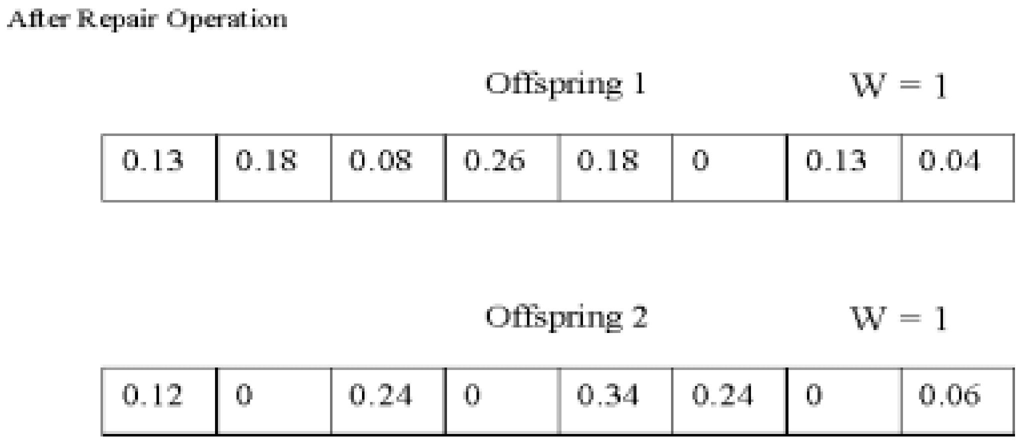
**Output**: Odata, output vector or chromosome, which is feasible

Variables:

LIND: # of genes in a chromosome

Sumx: Sum of gene values

```
1.  $Sumx = \sum_{i=1}^{LIND} Idata[i]$
2.  for i←1 to LIND
    3.  Odata[i]=Odata[i] / Sumx
4. return Odata
```


After Repair Operation

Offspring 1 W = 1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.13 | 0.18 | 0.08 | 0.26 | 0.18 | 0 | 0.13 | 0.04 |

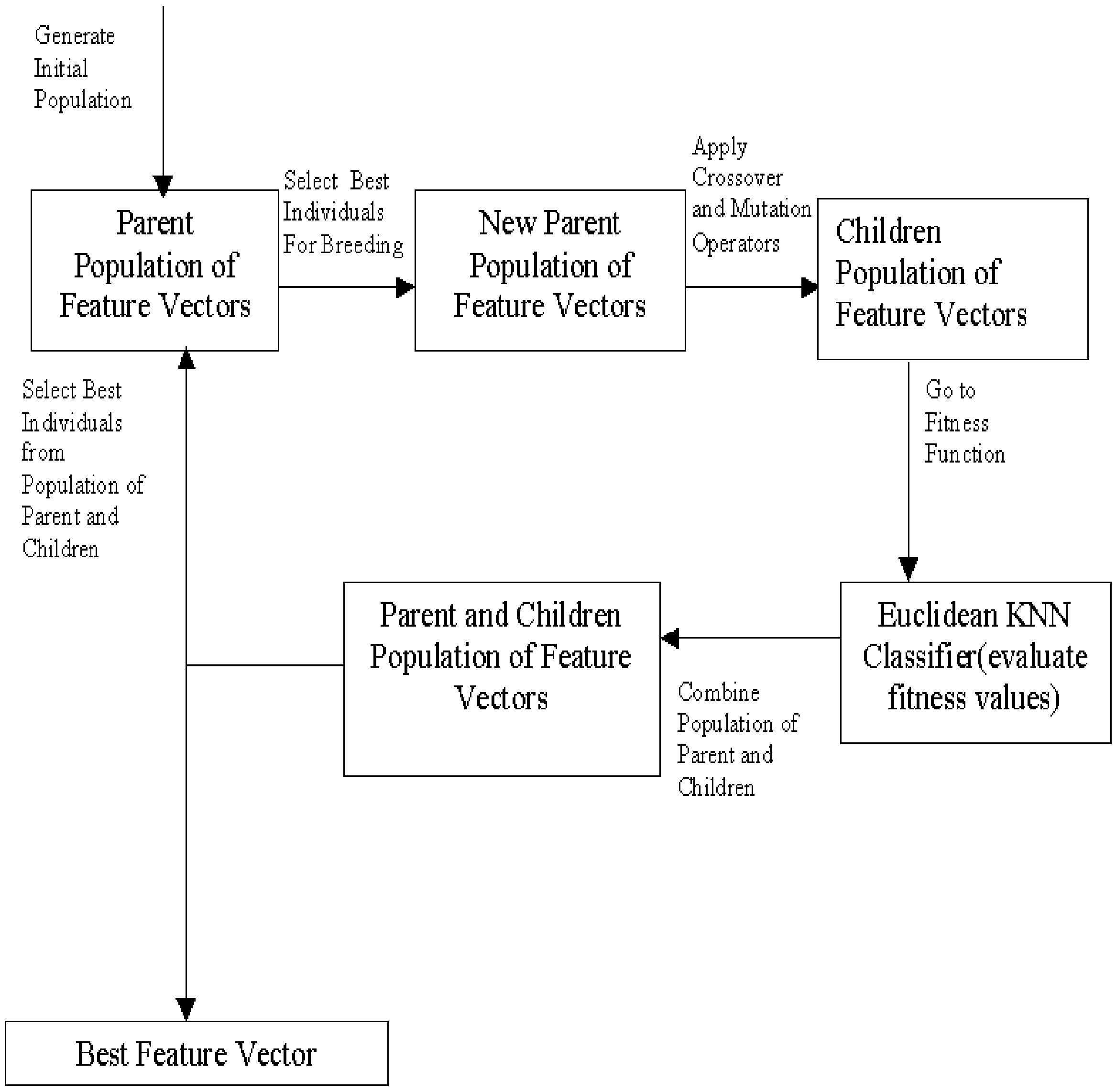
Offspring 2 W = 1

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.12 | 0 | 0.24 | 0 | 0.34 | 0.24 | 0 | 0.06 |

*Figure 11 Feasible Chromosomes After Repair Function*

## 4.6 GRaCCE Process Diagram

The overall execution processes of the binary-sGA and weighted-sGA are shown in Figure 14 and 15 respectively. As you can see the only difference between two algorithms is the Repair Function called for weighted-sGA.



Generate Initial Population
Parent Population of Feature Vectors
Select Best Individuals For Breeding
New Parent Population of Feature Vectors
Apply Crossover and Mutation Operators
Children Population of Feature Vectors
Go to Fitness Function
Euclidean KNN Classifier(evaluate fitness values)
Combine Population of Parent and Children
Parent and Children Population of Feature Vectors
Select Best Individuals from Population of Parent and Children
Best Feature Vector

*Figure 12 Binary feature subset selection using a simple genetic algorithm with GRaCCE*

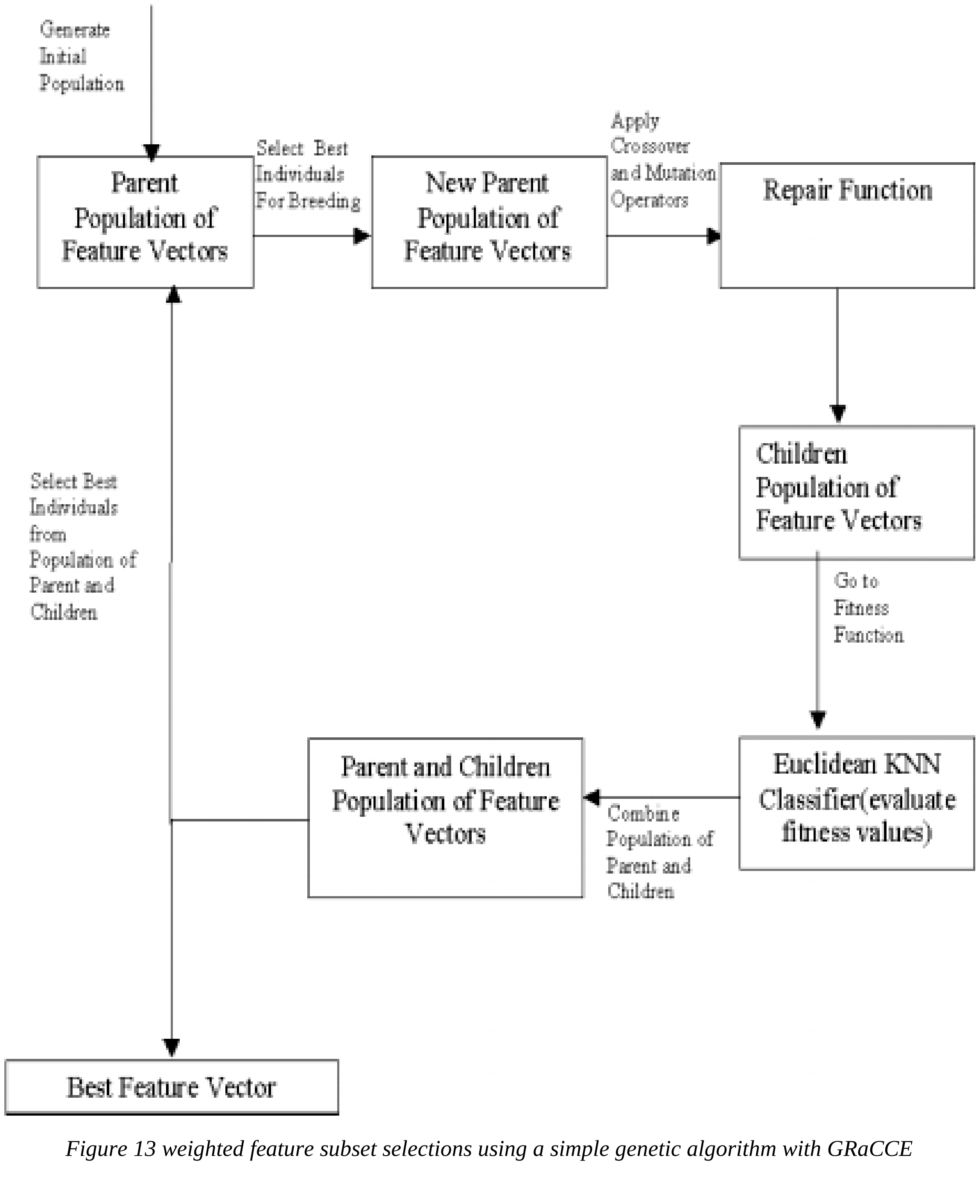

Generate Initial Population
Parent Population of Feature Vectors
Select Best Individuals For Breeding
New Parent Population of Feature Vectors
Apply Crossover and Mutation Operators
Repair Function
Children Population of Feature Vectors
Go to Fitness Function
Euclidean KNN Classifier(evaluate fitness values)
Combine Population of Parent and Children
Parent and Children Population of Feature Vectors
Select Best Individuals from Population of Parent and Children
Best Feature Vector

*Figure 13 weighted feature subset selections using a simple genetic algorithm with GRaCCE*

## 4.7 Objective Function And Fitness Function

The fitness function for our algorithm uses the results of the objective function to calculate the fitness values for each weight set in the weighted-sGA implementation or for each feature subset in the binary-sGA implementation. The objective function is simply the combination of the KNN classifier and the confusion matrix.

Since each gene represents a feature either as a weight or feature representation, if the value of a gene is zero, it means that the feature this gene is representing is not selected for the classification (i.e. original feature size is reduced.). Since each chromosome in the sGA represents a possible solution to feature weighting or future subset selection, KNN classifier must evaluate each chromosome's classification performance. For each chromosome in the population, KNN classifier is called. Each chromosome is multiplied with the standardized or normalized feature values in the training set or test set. This process is only called for one chromosome when test set is used and this chromosome is the best chromosome found in the training process of the KNN classifier. Multiplying the original instances and their feature values with either weight chromosomes or binary chromosomes produces a new feature space which has either scaled in each feature dimension or reduced in feature size. The instances are kept unchanged but their representations on the feature space are modified to speed the learning process of the KNN classifier and to decrease the classification error rate for each instance on the original data set. Each time KNN classifier chooses one instance from the modified temporary dataset as an unknown instance (Originally this instance is known but we are looking especially for the performance of feature weight or feature subset multiplied by this instance and the other instances in the dataset) and the others as known instances. Then KNN classifier calculates the distance from unknown instance to other known instances in the training dataset. Each times the distance between unknown instance and known instance is calculated, the calculated distance and the class of the known instance to whom distance calculated is stored in a nearest neighborhood list (nnlist). The nearest neighborhood list is a vector of structure type and defined using Microsoft Visual C++ 6.0 structure definition. The variables this structure has is given as follow:

```
struct Shared::nnlist{
int myclass;
real distvalue;
Subscript index;};
```

After the distance from unknown instance to other known instances is calculated and the nearest neighborhood list is built for all known instances, the nearest neighborhood list is searched for the k smallest distance values. Each time a minimum distance value is find in the nearest neighborhood list, the associated class value is read and the value of this class is incremented in the class (instance) counter matrix (ccnt). Each row in the class counter matrix represents a different class in the training set. For example if there were 3 different clusters in the training set, the size of the class counter matrix would be 3. When the minimum distance is found in the nearest neighborhood list, the class of this distance is read and it's associated class counter is incremented in the class counter matrix, the distance value for this distance is changed to infinity (i.e. a while which cannot be a minimum value) for not selecting the same nearest neighborhood list element again. After the nearest neighborhood list is read to find the minimum k-nearest neighbors to unknown instance and the class counter matrix for each k-nearest neighbor is modified according to their class values, the maximum value in the class counter matrix is found and the unknown instance is labeled as this class, which is the majority class in the class counter matrix. Since we are looking for the majority in the class counter matrix, to prevent the tie, k should be chosen with care. If there is a tie among classes, the smallest class is always chosen. But this bias toward the smallest class sometimes can produce inefficient results. Normally there is no unknown instance in the training set but by assuming each time a known instance as an unknown instance, and assuming the other instances as known instances we are evaluating our sGA chromosomes' performances in the classification process. A map vector is defined whose row size is equal to n where n is equal to the number of instances in the training dataset. Map vector is a column vector and each row in the map vector represents an instance in the training set. After calculating the class label for an unknown instance, the class label is written into the row of the map vector pointing to the unknown instance. As you can see this map vector holds the predicted class labels for each instance in the training set. The most important process of the KNN Classifier is building this map vector according to sGA weight or feature chromosomes multiplied by the instance feature values.

The next process in the classification is building the confusion matrix using the predicted class labels in the map vector and using the class labels in the original training set. Each row in the confusion matrix points to a different class labels in the original training set (0=class 0, 1- class 1....), and each column in the confusion matrix points to the predicted values of each class in the original set. Normally, if we have 100% classification accuracy which means the class of each instance in the training set is correctly classified, then the confusion matrix has non-zero values only on the diagonal, and zero value elsewhere. The non-zero values out of the diagonal of the confusion matrix show how many times a specific class is wrongly predicted during the classification process. For example, Let's assume that we have 4 classes from class1 to class4 and class 1 has 5 instances, class 2 has 10 instances, class 3 has 15 instances and class 4 has 20 instances. The results of a KNN classification for an sGA chromsome are shown in Figure 17. As you can see from Figure 15, the sum of each row in the confusion matrix is equal to the total number of instances for each class in the original set. The results show that out of 5 instances, one instance that belongs to class 1 is assigned to class 4 by a result of wrong prediction. The other for instances of the class 1 is correctly labeled with class 1. Out of 10 instances two instances that belong to class 2 are labeled as class 1 and class 3 respectively. Out of 15 instances one instance that belongs to class 3 is labeled as class 2 and the other 14 instances are correctly classified. Out of 20 instances two instances that belong to class 4 are labeled as class 3 and other 18 instances are correctly classified.

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **1** | 4 | 0 | 0 | 1 |
| **2** | 1 | 8 | 1 | 0 |
| **3** | 0 | 1 | 14 | 0 |
| **4** | 0 | 0 | 2 | 18 |

*Figure 14 Confusion Matrixes*

The last process in the objective function of the GRaCCE is the calculation of the percentage of the misclassification rate for the sGA chromosome for which the confusion matrix is built. The misclassification percentage is calculated as follows:

Misclassification rate = # of classes misclassified / total number of classes

Let's show this in our previous example,

Total number of classes = # of class 1 + # of class 2 + # of class 3 + # of class 4

= 5 +10 +15 +20

= 50

# Of classes misclassified = # of class 1 misclassified + # of class 2 misclassified

+ # of class 3 misclassified + # of class 4 misclassified

= 1 + 2 +1 +2

= 6

Misclassification rate = 6 / 50

Misclassification rate = 0.12

The value of misclassification rate says that the sGA chromosome whose performance is evaluated has an error rate of 12%. If there were other chromosomes in the population, which has less error rate than that of this chromosome, then the one, which has the least error rate, would be selected as the best chromosome (feature weight or feature subset) for this training data set.

The misclassification rate is calculated as a result of the objective function but the actual fitness function is a combination of the many objective functions. The fitness function must have the ability to combine many objective functions under only a one function since the sGA algorithm only tries to optimize the fitness values and there is only one fitness value for each chromosome in the sGA population.

The second objective function for our algorithms is the calculation of the total number of features, which are either weighted or selected. The reason for doing this is that if there are two weight chromosome that have the same misclassification rate, the second objective function ensures that the weight chromosome which weights less features and clean most features is selected as the best chromosome in the end of weighted-sGA algorithm. The objective function is applied to both weighted-sGA and binary-sGA algorithm without doing any changes.

The third objective function for our algorithm the calculation of the total cost for a weighted or selected feature subset. If two chromosomes have the same classification error rate, our choice is the one that has the least cost. The sGA algorithm uses this option only the features in a data set has measurement cost associated with them. If there is no measurement cost for any feature then this objective function is ignored in the fitness function and fitness value calculation.

Since the chromosomes are evaluated according to their fitness values, we combined each objective function under fitness function. The fitness function for the sGA algorithm is as follows:

$$\text{Fitness}(x) = \text{errorrate}(x) + \alpha*(\text{size}(x) + \text{cost}(x))$$

Where x = a chromosome in the sGA population and $\alpha$ is a user specified real value constant and $0<\alpha<1$.

If the features do not have any measurement cost than the result of cost (x) is equal to zero. The value for constant $\alpha$ is chosen as 0.0001 in our algorithm. Since we search for the best feature weight or feature subset in terms of classification accuracy, it is necessary to adjust $\alpha$ to a very small real value. We assume that the classification accuracy has the highest importance for us; we want our search be in favor of classification accuracy other than only reduction in the original feature size or the cost. High values for $\alpha$ constant changes the search space from searching for the best chromosome that has the least classification error rate to the searching for the best chromosome that has either less number of features or less cost. Our algorithm ensures that the selected chromosome has the least classification error rate in term of feature size and cost it is not guaranteed. But we know that the smallest size for a feature set would be zero and the cheapest cost for a feature set would be zero too, but this would be a solution to our problem since we need at least one feature to make classification on unknown instances.

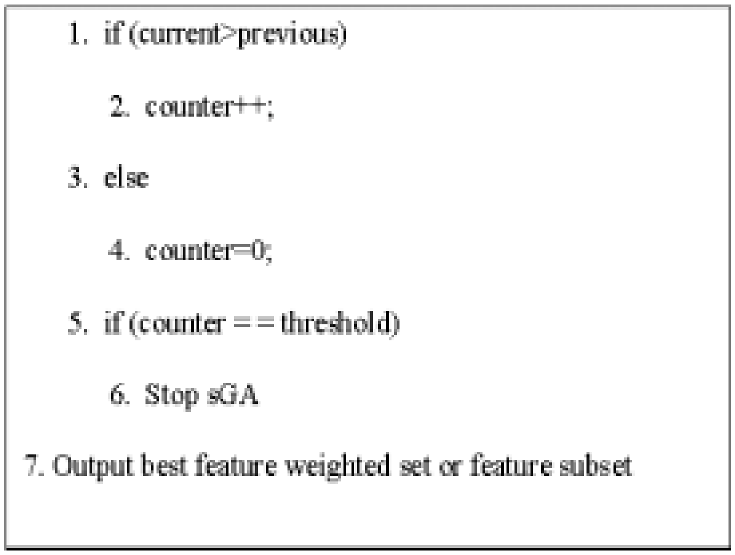
The sGA algorithm still searches for the chromosomes that has less feature size and less cost but in favor of classification error rate. It is not surprising to have some increase in the cost or the feature size during the execution of the sGA algorithm, this increases are only due to the reduction in the classification error rate.

## 4.8 Stop Criteria

A stop criteria algorithm is implemented to ensure that the classification error rate on the test data set does not increase while the classification error rate decrease on the train set. If the classification error rate increases on the test dataset while it decrease on the train dataset, we can say that there is a bias toward a specific solution (i.e. feature weights or feature subsets) which can no be generalized to test data set or any validation data set. But since the quality of a feature weight set or feature subset is judged according to their classification error rate on the test or validation datasets, it is not acceptable to have an increasing trend in terms of classification error rate on the test dataset.

**Input:** current, current classification error rate on test dataset; previous, previous classification error rate on test data set; counter, counter is incremented each time when the current classification error rate on test dataset is higher than the previous classification error on test dataset; threshold, userd specified value.

**Output:** False or True



```
    1. if (current>previous)
        2. counter++;
    3. else
        4. counter=0;
    5. if (counter = = threshold)
        6. Stop sGA
7. Output best feature weighted set or feature subset
```

*Figure 15 Stop Criteria Algorithm for Weighted-sGA and Binary-sGA Algorithms*

The stop criteria algorithm shown in Figure 17 guarantees that the trend of increase on the classification error rate for test data set cannot exceed the user specified threshold value. In our algorithm this threshold value is specified as 10.

## 4.9 Chapter Summary

In this chapter we explained the execution process of our sGA algorithm and loading of the datasets onto GRaCCE environment, encoding scheme we have chosen in our sGA algorithm, the fitness function used to evaluate the quality of each individual in the population, repair algorithm we used to modify the infeasible chromosomes and we explained the stopping criteria we implemented for more accurate solution to out problem area.

# *V. Design Of Experiments and Analysis*

## 5.1 Introduction

This chapter discusses the goal expected from the experiments, the performance metrics, experiment parameters, the datasets used in the experiments and the method of analysis for the results.

## 5.2 Goal of the Experiment

The goal of this study is to analyze the performance of GRaCCE in terms of classification accuracy by using feature weighting technique and binary feature subset selection technique. These two techniques are compared in terms of their classification accuracy rate. The overall performance of the GraCCE is also compared with those of other Classification algorithms. The system under test is GRaCCE and the component under test is weigted-sGA and binary-sGA algorithms.

## 5.3 System Services

The service we expect from our system is to classify unknown instances, calculate the distances of each feature to other features and calculate the classification accuracy rates for data sets presented to the system where the system is GRaCCE. The possible outcomes that we expect from GRaCCE is to exhaust the number of GA generations specified at the beginning of feature weighting and selection process, or to stop execution at an early stage when the specified stopping criteria is reached-in our case defined as having worse classification rate for test data set when the number of generation reaches 10. This value is empirical and can be changed according to user request.

## 5.4 Performance Metric

The only performance metric we used to evaluate the performance of the GRaCCE is the classification error rate found by GRaCCE for different data sets. But there are other metrics such as distance metrics used by the KNN classifier and in that case this distance metric is Euclidean distance to calculate the distances between features on the same dimension. Another metric is the "cost" metric for measuring a feature to describe a class or pattern. We did not use this metric since we do not know the cost of measurement for all features used in the experiments.

## 5.5 Experiment Parameters

There are two types of parameters in our experiments that need to be explained. These parameters are hardware and system parameters. The hardware parameters are the CPU clock rate, size of memory, type of network interconnection and the operating system used during the experiments. The system parameters are the selection mechanism of GA for breeding new individual, the probability of crossover and mutation operators, types of crossover and mutation operators used during experiments and the value of K for KNN classifier and the percentages of the training and test data sets. The number of individuals in the GA population and the number of generations that the GA has to be run are the other system parameters.

| The Experiment Environment is shown in Table 4.The Experiment Environment | |
|---|---|
| CPU | Intel Pentium 4 2GHz |
| RAM Memory | 1024MB |
| Operating System | Microsoft Windows 2000 Professional Edition |
| Programming Language | Microsoft Visual C++ Version 6.0 |

*Table 4 Experiment Environments*

The sGA Algorithm parameters that we used in our experiments are explained in Table 5.

| | |
|---|---|
| # Of Generation | 60 |
| Population Size | 30 |
| Crossover probability | 1.0 |
| Mutation probability | 0.1 |
| The value of K | 1 |
| The percentage of training set | 0.70 |
| The percentage of test set | 0.30 |

*Table 5 The Value of GRaCCE Parameters For the Experiments*

The values shown in Table are decided upon many preliminary experiments and kept constant at the time of experiments.

As a selection operator, "linear ranking" method is used. The crossover and mutation operators used in the experiments are two-point crossover operator and exchange mutation respectively for weighted-sGA experiments and binary two-point crossover operator and inversion mutation operators for binary-sGA are used. Since it is not possible to invert a real value in the case of inversion mutation, different mutation operators are implemented for these two algorithms.

We used 15 real world-datasets, which have different number of characteristics. Some datasets have small number of features and classes (e.g. Iris dataset has only 4 features and 2 classes), Some datasets have large number of features and classes (e.g. Pilot dataset has 108 features). We did not make any assumption about the distribution of datasets. We designed a full factorial experiment with 10 replications for each dataset. Since we use 15 datasets it gives us 15*10=150 experiments for only one algorithm. Since we are evaluating the performance of two different algorithm, for each algorithm we design 240 experiments which gives us a total number of 2*150=300 experiments.

We used two different evaluation techniques together, which are statistical analysis and simulation. The workload used for the system was changed according to the size of the data sets used in our experiments.

Our research hypothesis is that in addition to feature subset selection, assigning weights to the selected features increases the classification accuracy than that of feature subset selection which assigns weights to features either zero or one instead of real values. The null hypothesis is the negation of our alternative hypothesis that is there will be no increase on classification accuracy via feature weighting and feature selection. According to statements given we can formulate our hypothesis as follows:

$H_1$: Classification accuracy of feature weighting $\geq$ Classification accuracy of feature subset selection

$H_0$: Classification accuracy of feature weighting < Classification accuracy of feature subset selection.

$H_1$: $\mu_{\text{feature weighting}} \geq \mu_{\text{feature selection}}$

$H_0$: $\mu_{\text{feature weighting}} < \mu_{\text{feature selection}}$

Where

$\mu$ = the mean of classification accuracy.

We use significance testing technique to decide whether we can reject the null hypothesis or not. For this purpose we calculate the sample mean of our experiments for classification accuracy and the standard deviation. Let's give an example:

Let's say for the Fisher's IRIS data set the mean of the previous 16 experiments for classification accuracy is 94.7 with a standard deviation of 3.2. The current study has a mean of 96.4 for classification accuracy with a standard deviation of 0.72. To see if there is enough difference to cause us to reject $H_0$, we find the P value for the test. That is, we compute the probability of observing a sample mean of 96.4 or larger if $\mu = 94.7$ and $\sigma = 3.2$, then the test statistic X is, by Central Limit Theorem, at least approximately normally distributed with mean $\mu = 94.7$ and standard deviation $\sigma/\sqrt{n} = 3.2/4 = 0.8$. Therefore

$P[X \geq 96.4 \mid \mu = 94.7 , \sigma = 3.2 ] = 1\text{-}P[Z \leq 2.125] = 1\text{-}.9832 = 0.0168$ from Standard Normal Table[28]

There are two explanations for this very small probability. The null hypothesis is true and we have observed a very rare sample that by chance has a larger sample mean; the null hypothesis is not true and the new classification technique which is the combination of the feature weighing and feature selection, resulted in a higher mean of classification accuracy. So we shall reject $H_0$, and report that the P value of our test id 0.0168.

Another technique that we used to analyze our results is zero mean testing [27]. A common use of confidence intervals is to check if a measured value is significantly different from zero. When comparing a random measurement with zero, the statements have to be made probabilistically, that is, at a desired level of confidence. If the measured value passes our test of difference with a probability greater than or equal to the specified level of confidence, 100(1 - $\alpha$)%, then the value is significantly different from zero. The test consists of determining a confidence interval and simply checking if the interval includes zero. If the confidence interval does not include zero then we can say that one system is superior to the other, but if confidence interval includes zero, and therefore, the measured values are not significantly different from zero, so the compared systems are not significantly different at that confidence level.

The experiments we report here used real world data sets. We obtained the real-world data sets from the machine-learning repository at the University of California, Irvine (http://www.ics.uci.edu/AI/ML/MLDBRepository.html)

Our objective with real-world data sets was to compare the performance of weighted sGA in terms of classification accuracy with that of binary-sGA. Some medical data sets include measurement costs for the features, but most sets lack this information. Thus, our experiments focused on only identifying a minimal subset of features and their associated weights to yield high classification accuracy KNN classifier using sGA algorithm for all data sets. But further studies can take these costs into consideration and develop a new fitness function that evaluates also the cost of each feature in the classification problem and selects the individuals which provides not only high classification accuracy but also less cost for the classification of unknown patterns.

Total number of data sets used in the experiments is 15 and 13 of these datasets are shown in Table 4, and each data set has similar and different characteristics. The FLIR (Scud) and PILOT datasets are unclassified datasets used to perform target recognition of SCUD missiles and behavior recognition of PILOTS respectively. Some data sets include numeric values, some data sets include nominal values, and some datasets include both values. A set of data is said to be nominal if the values/ observations belonging to it can be assigned a code in the form of a number where the numbers are simply labels. You can count order or measure nominal data. For example, in a data set males could be coded as 0, females as 1; marital status of an individual could be coded as Y if married, N if single. Also some datasets like hepatitis and vote datasets have missing values in their some features, which has to be taken into consideration while evaluating their classification accuracy rate.

10 Experiments are conducted for each algorithm (weighted-sGA and binary-sGA) on training set to find the classification accuracy of selected feature set and also the classification accuracy rate of this selected feature set is calculated on test set to see the effect of reduced set, since the quality of a reduced feature set is usually judged upon the test set or unknown future examples.

| DATA SET | DESCRIPTION | SIZE | INPUT FEATURES | FEATURE TYPE | OUTPUT CLASSES |
|---|---|---|---|---|---|
| Cancer | Breast Cancer | 699 | 9 | Numeric | 2 |
| Flag | Flag Database | 194 | 28 | Numeric, nominal | 8 |
| Glass | Glass identification | 214 | 9 | Numeric | 6 |
| Hepatitis | Hepatitis domain | 155 | 19 | Numeric, nominal | 2 |
| Iono | Ionosphere structure | 351 | 34 | Numeric | 2 |
| Liver | Liver disorders | 345 | 6 | Numeric | 2 |
| Pima | Pima Indians diabetes | 768 | 8 | Numeric | 2 |
| Sonar | Sonar Classification | 208 | 60 | Numeric | 2 |
| Soybean | Large soybean | 307 | 35 | Nominal | 19 |
| Votes | House votes | 435 | 16 | Nominal | 2 |
| Wine | Wine recognition | 178 | 13 | Numeric | 3 |
| Tic-tac-toe | Tic-tac-toe game | 958 | 9 | Nominal | 2 |
| Iris | Iris Plant Database | 150 | 4 | Numeric | 3 |

*Table 6 Data Sets used in the experiments*

## 5.6 Data Sets Used In the Experiments

**Iris**: This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. Number of instances is 150 (50 in each of three classes). There are 4 numeric predictive attributes and the class in the database. The attributes are sepal length in cm, sepal width in cm, petal length in cm, petal width in cm and the classes are Iris Setosa, Iris Versicolour, and Iris Virginica. There are no missing attribute values.

| Summary Statistics | | | | | |
|---|---|---|---|---|---|
| | Min | Max | Mean | StdDev | Class Correlation |
| Sepal length | 4.3 | 7.9 | 5.84 | 0.83 | 0.7826 |
| Sepal width | 2.0 | 4.4 | 3.05 | 0.43 | -0.4194 |
| Petal length | 1.0 | 6.9 | 3.76 | 1.76 | 0.9490(high!) |
| Petal width | 0.1 | 2.5 | 1.20 | 0.76 | 0.9565 (high!) |

*Table 7 Summary Statistics For Fisher's Iris Database*

As you can see from Table 7, third and fourth features are highly correlated and the feature subset selection algorithm has to select these two features for classification of the Iris plant. In Figure 15, the original class distribution is shown using the feature 1, feature 2 and feature 3, feature 4. There are 33.3% from each of 3 classes.

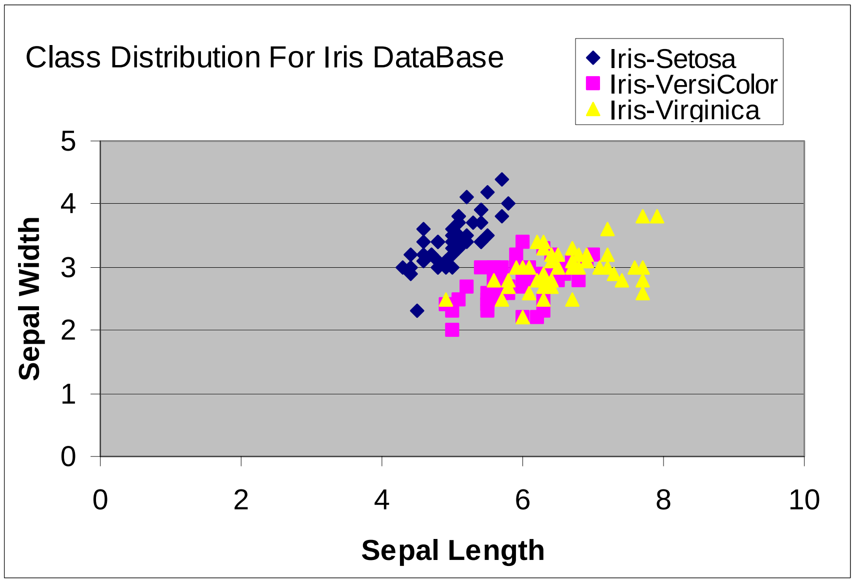


*Figure 16 Class Distribution For Iris Database (Feature 1, Feature 2)*

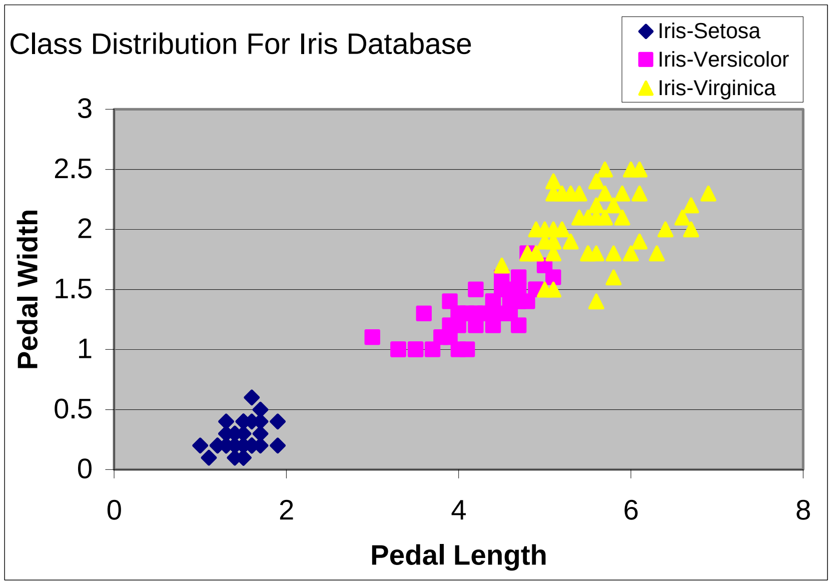


*Figure 17 Class Distributions for Iris Database (Feature 3, Feature 4)*

**Glass:** The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence...if it is correctly identified! There are 214 instances and 10 attributes (including an id#) plus the class attribute and all attributes are continuously valued. The 10 attributes are, Id number from 1 to 214, refractive index, Sodium, Magnesium, Aluminum,

Silicon, Potassium, Calcium, Barium, Iron. There are 6 classes (types of glasses) as building_windows_float_processed,
building_windows_non_float_processed,vehicle_windows_float_processed,
vehicle_windows_non_float_processed (none in this database), containers, tableware, and headlamps. There are no missing attribute values. There are 163 window glasses and 51 non-window glasses. The detailed information about glass distribution can be found from the machine-learning repository at the University of California, Irvine.

**Tic-tac-toe:** This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where “x” is assumed to have played first. The target concept is “win for x” (i.e., true when “x” has one of 8 possible ways to create a “three-in-a-row”). Interestingly, this raw database gives a stripped-down decision tree algorithm (e.g., ID3) fits. However, the rule-based CN2 algorithm, the simple IB1 instance-based learning algorithm, and the CITRE feature-constructing decision tree algorithm perform well on it. Number of Instances is 958 (legal tic-tac-toe endgame boards), number of attributes is 9 (each corresponding to one tic-tac-toe square) and for an attribute “x” means player x has taken, “o” means player o is taken, and “b” means blank. In our algorithm, we coded the attribute values” x,o,b” with numeric values 1,0,-1 respectively. Since our algorithm accepts only numeric values, this conversion is necessary to have feasible solutions. The attribute information is given as follows:

1. top-left-square: {x,o,b}
2. top-middle-square: {x,o,b}
3. top-right-square: {x,o,b}
4. middle-left-square: {x,o,b}
5. middle-middle-square: {x,o,b}
6. middle-right-square: {x,o,b}
7. bottom-left-square: {x,o,b}
8. bottom-middle-square: {x,o,b}
9. bottom-right-square: {x,o,b}
10. Class: {positive,negative}

There are no missing attribute values. About 65.3% of instances are positive (i.e., wins for "x"). There are some reported classification accuracy rate for this data set when 70% used as training set and 30% used as test set as follows: NewID: 84.0%, CN2: 98.1%, MBRtalk: 88.4%, IB1: 98.1%, IB3: 82.0%, IB3-CI: 99.1%

**Soybean:** There are 19 classes, only the first 15 of which have been used in prior work. The folklore seems to be that the last four classes are unjustified by the data since they have so few examples. There are 35 categorical attributes, some nominal and some ordered. The value “dna”" means does not apply. The values for attributes are encoded numerically, with the first value encoded as ``0," the second as

“1”and so forth. An unknown values is encoded as ”?’’. There are 307 instances, 35 features (all have been nominalized) There are 19 classes as diaporthe-stem-canker, charcoal-rot, rhizoctonia-root-rot, phytophthora-rot, brown-stem-rot, powdery-mildew, downy-mildew, brown-spot, bacterial-blight, bacterial-pustule, purple-seed-stain, anthracnose, phyllosticta-leaf-spot, alternarialeaf-spot, frog-eye-leaf-spot, diaporthe-pod-&-stem-blight, cyst-nematode, 2-4-d-injury, herbicide-injury.

The features are:

| | |
|---|---|
| 1. date: | april,may,june,july,august,september,october,?. |
| 2. plant-stand: | normal,lt-normal,?. |
| 3. precip: | lt-norm,norm,gt-norm,?. |
| 4. temp: | lt-norm,norm,gt-norm,?. |
| 5. hail: | yes,no,?. |
| 6. crop-hist: | diff-lst-year,same-lst-yr,same-lst-two-yrs, same-lst-sev-yrs,?. |
| 7. area-damaged: | scattered,low-areas,upper-areas,whole-field,?. |
| 8. severity: | minor,pot-severe,severe,?. |
| 9. seed-tmt: | none,fungicide,other,?. |
| 10. germination: | 90-100%,80-89%,lt-80%,?. |
| 11. plant-growth: | norm,abnorm,?. |
| 12. leaves: | norm,abnorm. |
| 13. leafspots-halo: | absent,yellow-halos,no-yellow-halos,?. |
| 14. leafspots-marg: | w-s-marg,no-w-s-marg,dna,?. |
| 15. leafspot-size: | lt-1/8,gt-1/8,dna,?. |
| 16. leaf-shread: | absent,present,?. |
| 17. leaf-malf: | absent,present,?. |
| 18. leaf-mild: | absent,upper-surf,lower-surf,?. |
| 19. stem: | norm,abnorm,?. |
| 20. lodging: | yes,no,?. |
| 21. stem-cankers: | absent,below-soil,above-soil,above-sec-nde,?. |
| 22. canker-lesion: | dna,brown,dk-brown-blk,tan,?. |
| 23. fruiting-bodies: | absent,present,?. |
| 24. external decay: | absent,firm-and-dry,watery,?. |
| 25. mycelium: | absent,present,?. |
| 26. int-discolor: | none,brown,black,?. |
| 27. sclerotia: | absent,present,?. |
| 28. fruit-pods: | norm,diseased,few-present,dna,?. |

|  |  |
|---|---|
| 29. fruit spots: | absent,colored,brown-w/blk-specks,distort,dna,?. |
| 30. seed: | norm,abnorm,?. |
| 31. mold-growth: | absent,present,?. |
| 32. seed-discolor: | absent,present,?. |
| 33. seed-size: | norm,lt-norm,?. |
| 34. shriveling: | absent,present,?. |
| 35. roots: | norm,rotted,galls-cysts,?. |

**Zoo :** A simple database containing 17 Boolean-valued attributes. The "type" attribute appears to be the class attribute. Here is a breakdown of which animals are in which type: (I find it unusual that there are 2 instances of "frog" and one of "girl"!). There are 7 classes as:

**Class 1** (41) aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby,wolf

**Class 2** (20) chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren

**Class 3** (5) pitviper, seasnake, slowworm, tortoise, tuatara

**Class 4** (13) bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna

**Class 5** (4) frog, frog, newt, toad

**Class 6** (8) flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp

**Class 7** (10) clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

There are 101 instrances and 18 features (animal name, 15 boolean attributes, 2 numeric). There are no missing values and he attributes are:

|  |  |
|---|---|
| 1. animal name: | Unique for each instance |
| 2. hair | Boolean |
| 3. feathers | Boolean |
| 4. eggs | Boolean |
| 5. milk | Boolean |
| 6. airborne | Boolean |
| 7. aquatic | Boolean |
| 8. predator | Boolean |
| 9. toothed | Boolean |
| 10. backbone | Boolean |

| | |
|---|---|
| 11. breathes | Boolean |
| 12. venomous | Boolean |
| 13. fins | Boolean |
| 14. legs | Numeric (set of values: {0,2,4,5,6,8}) |
| 15. tail | Boolean |
| 16. domestic | Boolean |
| 17. catsize | Boolean |
| 18. type | Numeric (integer values in range [1,7]) |

**Hepatitis:** There are 155 instances and 20 features including class attribute. There are 2 classes as DIE or LIVE. The features are:

1. Class: DIE, LIVE
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
3. SEX: male, female
4. STEROID: no, yes
5. ANTIVIRALS: no, yes
6. FATIGUE: no, yes
7. MALAISE: no, yes
8. ANOREXIA: no, yes
9. LIVER BIG: no, yes
10. LIVER FIRM: no, yes
11. SPLEEN PALPABLE: no, yes
12. SPIDERS: no, yes
13. ASCITES: no, yes
14. VARICES: no, yes
15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
17. SGOT: 13, 100, 200, 300, 400, 500,
18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
20. HISTOLOGY: no, yes

There are many missing values.

**Votes:** This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

There are 435 instances (267 democrats, 168 republicans). There are 16 features plus class attribute. The attributes are:

1. Class Name: 2 (democrat, republican)
2. handicapped-infants: 2 (y,n)
3. water-project-cost-sharing: 2 (y,n)
4. adoption-of-the-budget-resolution: 2 (y,n)
5. physician-fee-freeze: 2 (y,n)
6. el-salvador-aid: 2 (y,n)
7. religious-groups-in-schools: 2 (y,n)
8. anti-satellite-test-ban: 2 (y,n)
9. aid-to-nicaraguan-contras: 2 (y,n)
10. mx-missile: 2 (y,n)
11. immigration: 2 (y,n)
12. synfuels-corporation-cutback: 2 (y,n)
13. education-spending: 2 (y,n)
14. superfund-right-to-sue: 2 (y,n)
15. crime: 2 (y,n)
16. duty-free-exports: 2 (y,n)
17. export-administration-act-south-africa: 2 (y,n)

There are a few missing values.

**Flag:** This data file contains details of various nations and their flags. In this file the fields are separated by spaces (not commas). With this data you can try things like predicting the religion of a country from its size and the colors in its flag. 10 attributes are numeric-valued. The remainders are either Boolean or nominal-valued.

There are 194 instances and 30 features. The attributes are:

| | |
|---|---|
| 1. name | Name of the country concerned |
| 2. landmass | 1=N.America, 2=S.America, 3=Europe, 4=Africa, 4=Asia, 6=Oceania |
| 3. zone | Geographic quadrant, based on Greenwich and the Equator 1=NE, 2=SE, 3=SW, 4=NW |
| 4. area | in thousands of square km |

5. population in round millions

6. language 1=English, 2=Spanish, 3=French, 4=German, 5=Slavic, 6=Other Indo-European, 7=Chinese, 8=Arabic, 9=Japanese/Turkish/Finnish/Magyar, 10=Others

7. religion 0=Catholic, 1=Other Christian, 2=Muslim, 3=Buddhist, 4=Hindu, 5=Ethnic, 6=Marxist, 7=Others

8. bars Number of vertical bars in the flag

9. stripes Number of horizontal stripes in the flag

10. colours Number of different colours in the flag

11. red 0 if red absent, 1 if red present in the flag

12. green same for green

13. blue same for blue

14. gold same for gold (also yellow)

15. white same for white

16. black same for black

17. orange same for orange (also brown)

18. mainhue predominant colour in the flag (tie-breaks decided by taking the topmost hue, if that fails then the most central hue, and if that fails the leftmost hue)

19. circles Number of circles in the flag

20. crosses Number of (upright) crosses

21. saltires Number of diagonal crosses

22. quarters Number of quartered sections

23. sunstars Number of sun or star symbols

24. crescent 1 if a crescent moon symbol present, else

25. triangle 1 if any triangles present, 0 otherwise

26. icon 1 if an inanimate image present (e.g., a boat), otherwise 0

27. animate 1 if an animate image (e.g., an eagle, a tree, a human hand) present, 0 otherwise

28. text 1 if any letters or writing on the flag (e.g., a motto or slogan), 0 otherwise

29. topleft colour in the top-left corner (moving right to decide tie-breaks)

30. botright Colour in the bottom-left corner (moving left to decide tie-breaks)

Table 8 shows the full list datasets that can be obtained from Irvine Repository.

| DATASET | DESCRIPTION | # OF INSTANCE | # OF FEATURES | # OF CLASSES |
|---|---|---|---|---|
| Annealing | Annealing Data | 798 | 38 | 6 |
| Audiology | Audiology Database | 226 | ??? | 24 |
| Auto | Auto Imports Database | 205 | 26 | - |
| Bridges | Pittsburgh bridges | 108 | 13 | - |
| CPU Performance | Relative CPU Performance Data | 209 | 10 | - |
| Echocardiagram | Echocardiogram Data | 132 | 13 | 2 |
| Heart | Heart Disease Databases | 303 (Cleveland database) | 76 | - |
| Image | Image Segmentation data | 210 + 2100 | 19 | 7 |
| Kinship | Kinship domain (relational) | 104 | 12 | - |
| Lenses | Database for fitting contact lenses | 24 | 4 | 3 |
| Letter Recognition | Letter Image Recognition Data | 20000 | 17 | 26 |
| Liver | BUPA liver disorders | 345 | 7 | - |
| Mushroom | Mushroom Database | 8124 | 22 | 2 |
| Shuttle Landing | Space Shuttle Auto landing Domain | 15 | 7 | 2 |
| Thyroid | Thyroid disease | 9172 | 29 | 6 |
| University | Lebowitz's Universities Database | 285 | 17 | - |
| Waveform | Waveform Database Generator (written in C) | 5000 | 21 | 3 |
| Protein | Secondary Structure of Globular Proteins | - | - | - |

*Table 8 Real-World Datasets in Irvine Repisotory*

## 5.7 Chapter Summary

In this chapter we explained our design of experiments and the analysis. We gave our justification for the experiment parameters and sGA parameters such as crossover and mutation probability, method of selection etc. We also explained the real-world datasets used in our experiments.

The analysis and interpretation of our results is given in Chapter 6.

# *VI. EXPERIMENT RESULTS*

## 6.1 Introduction

This chapter presents the results of our experiments for real-world datasets using weighted-sGA and binary-sGA. For the calculation of our results the average of 15 experiments are calculated for weighted sGA and binary sGA. The plots show the performance of each algorithm for each real-world dataset in terms of their classification accuracy obtained after 10 experiments for each dataset.

Figure 18 shows the average classification accuracy rate of 10 experiments for Cancer dataset on training set. The resulting classification accuracy rate for weighted classification is better than that of binary classification on Cancer training set. According to Figure 18, it is clear that binary-sGA converges rapidly and the convergence is reached around 14th generation but on the other hand the weighted-sGA converges slower than binary-sGA and according to Figure 18, weighted-sGA converges aroung 55th generation. This result shows that weighted-sGA has more diverse population and the crossover and mutation operators produce better individuals in terms of classification accuracy rate.

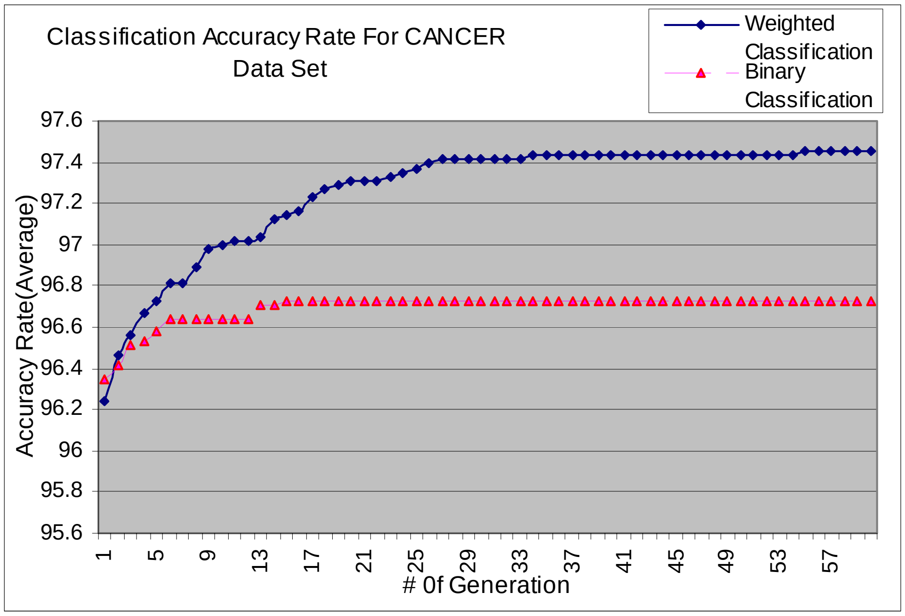


*Figure 18 Classification Accuracy Rate on Cancer Train Set For Weighted-sGA andBinary-sGA*

Figure 19 shows the average classification accuracy rate for test set of the Cancer dataset. According to Figure 19, weighted classification has also better classification accuracy than binary classification on the test data.

But since the number of examples in test set is less than that of training set, and the feature weights and feature subsets are found according to training set, the classification accuracy rate on test set is worse than that of training set, and the trend of convergence is also different. As you can see from Figure 19, there is no convergence and different feature weights and subsets have different classification accuracy rate and the increasing trend of classification accuracy rate toward the end of the sGA generation as in training set is not applicable to the test set but still we have good classification accuracy rate on test set since the resulting classification accuracy rate is higher than the beginning classification accuracy rate. WE can say that the feature weights and feature subset found by sGA algorithm can be generalized to future unknown examples.

You can see the results for Hepatitis, Iris, Liver, Sonar, Pilot in Figures 20, 21, 22 ,23, 24, 25 ,26, 27, 28, 29, respectively.

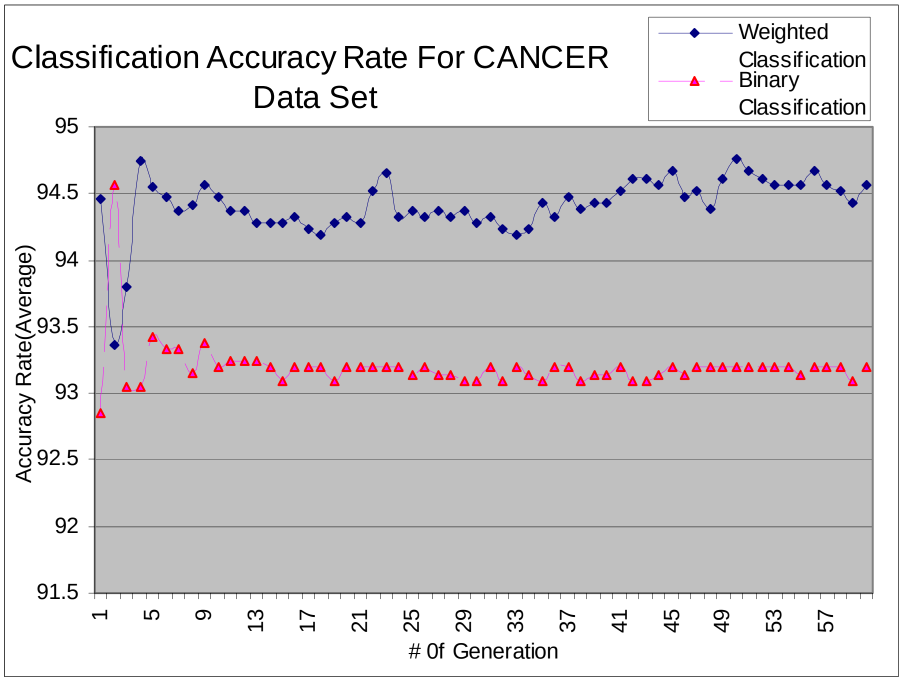


*Figure 19 Classification Accuracy Rate on Cancer Test Set For Weighted-sGA and Binary-sGA*

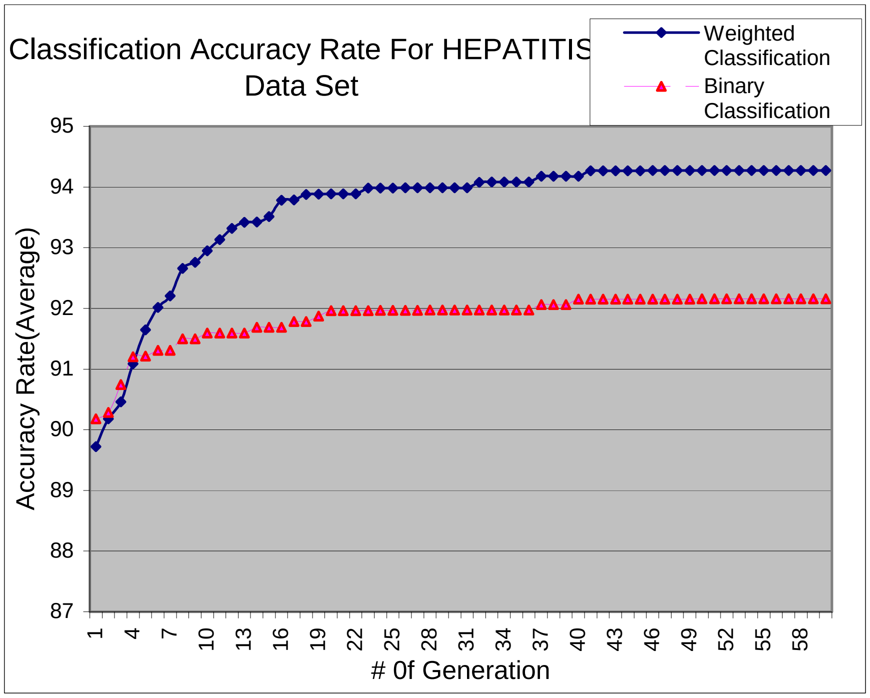


*Figure 20 Classification Accuracy Rate on Hepatitis Train Set For Weighted-sGA and binary-sGA*

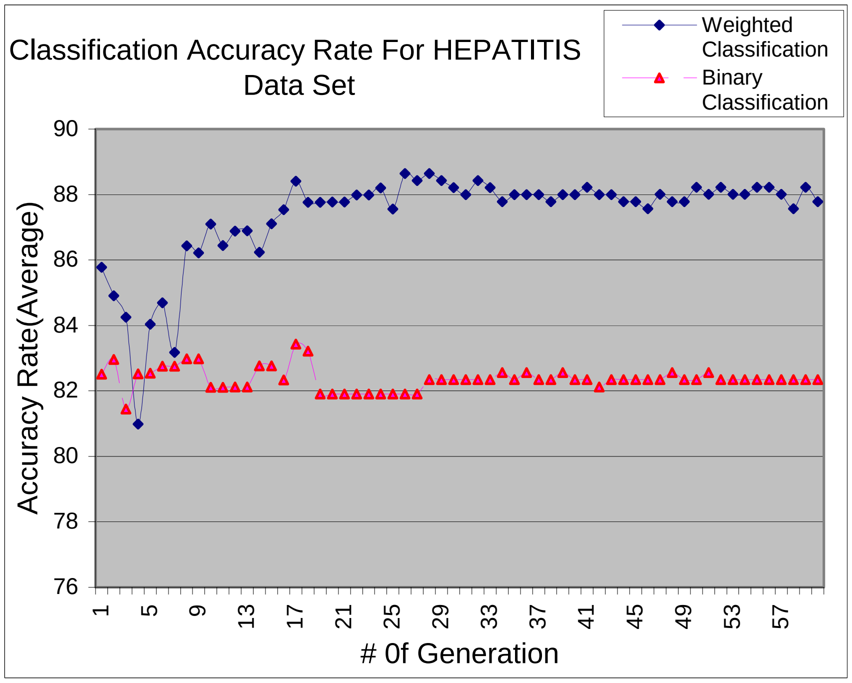


*Figure 21 Classification Accuracy Rate on Hepatitis Test Set For Weighted-sGA and Binary-sGA*

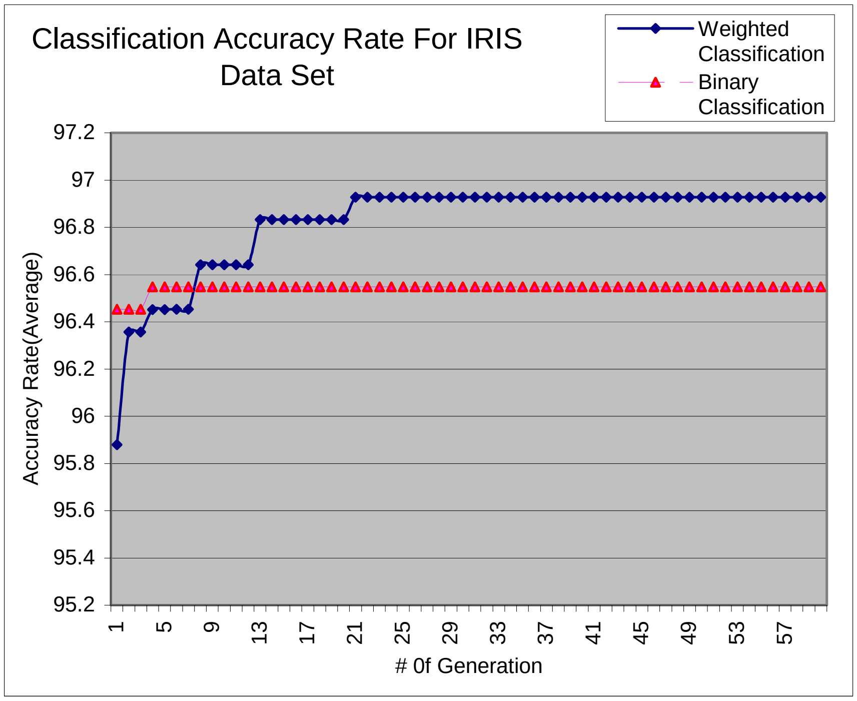


*Figure 22 Classification Accuracy Rate on Iris Train Set For Weighted-sGA and Binary-sGA*

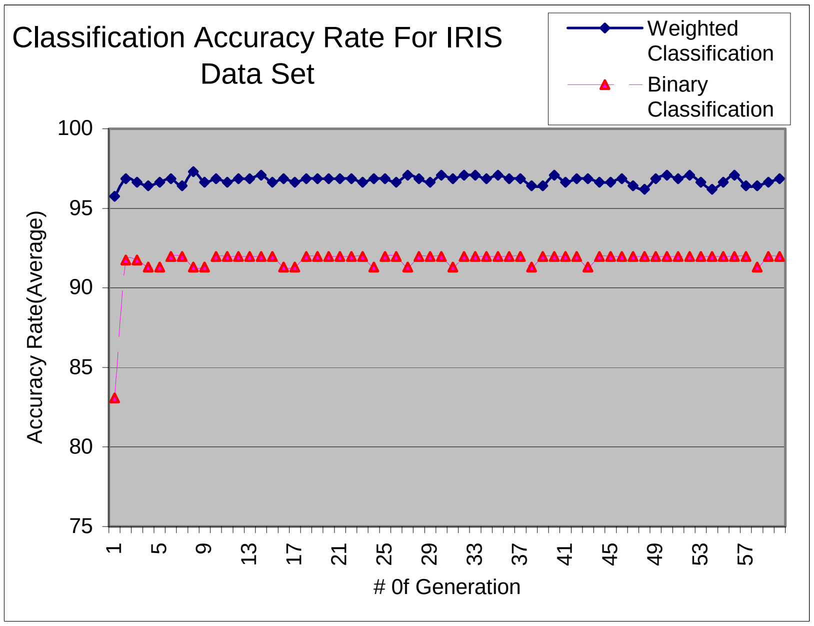


*Figure 23 Classification Accuracy Rate on Iris Test Set For Weighted-sGA and Binary-sGA*

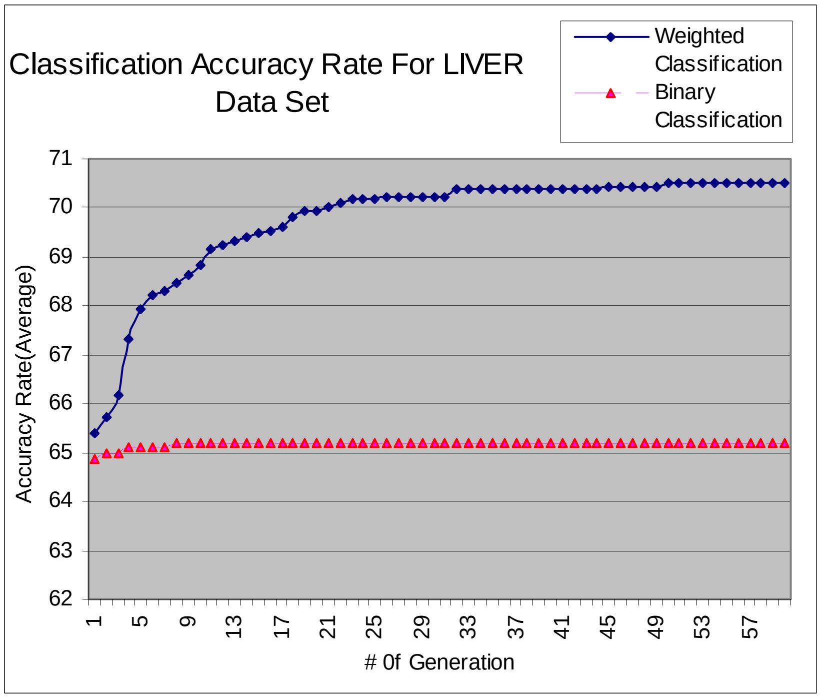


*Figure 24 Classification Accuracy Rate on Liver Train Set For Weighted-sGA and Binary-sGA*

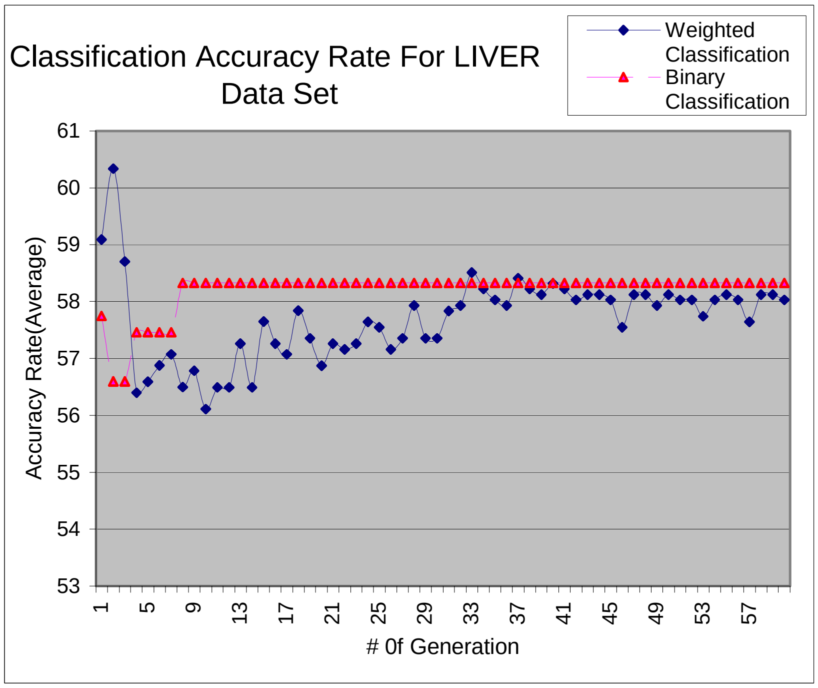


*Figure 25 Classification Accuracy Rate on Liver Test Set For Weighted-sGA and Binary-sGA*

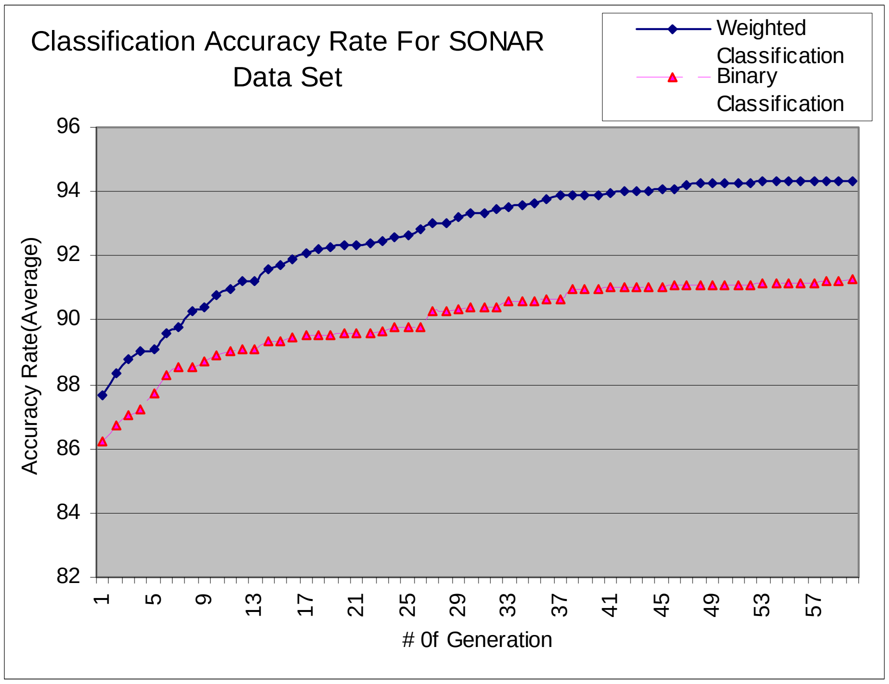


*Figure 26 Classification Accuracy Rate on Sonar Train Set For Weighted-sGA and Binary-sGA*

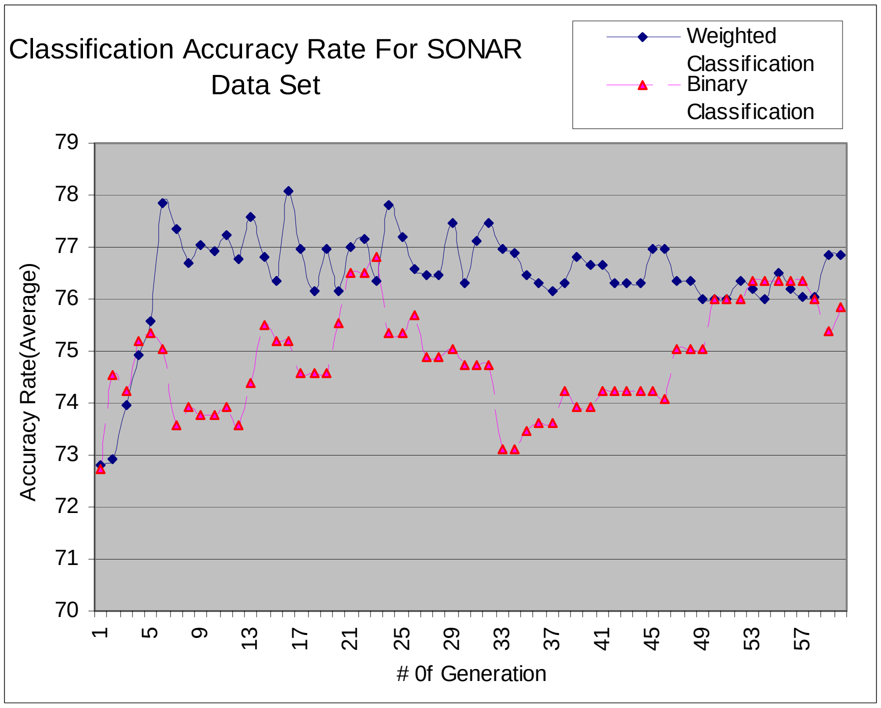


*Figure 27 Mean Classification Accuracy Rate on Sonar Test Set For Weighted-sGA and binary-sGA*

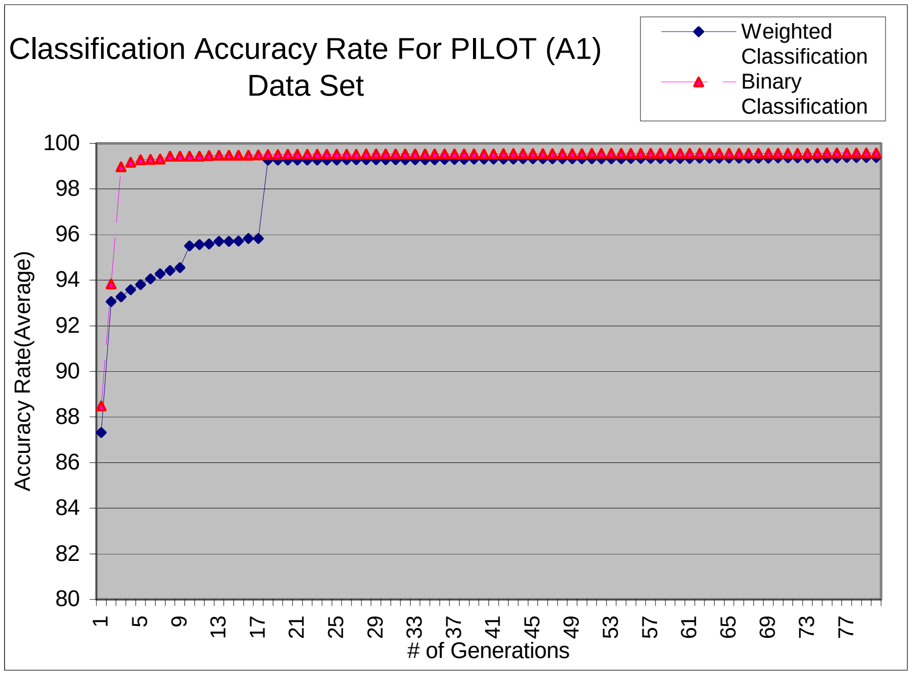


*Figure 28 Classification Accuracy Rate on Pilot (A1) Train Set over For Weighted-sGA and binary-sGA*

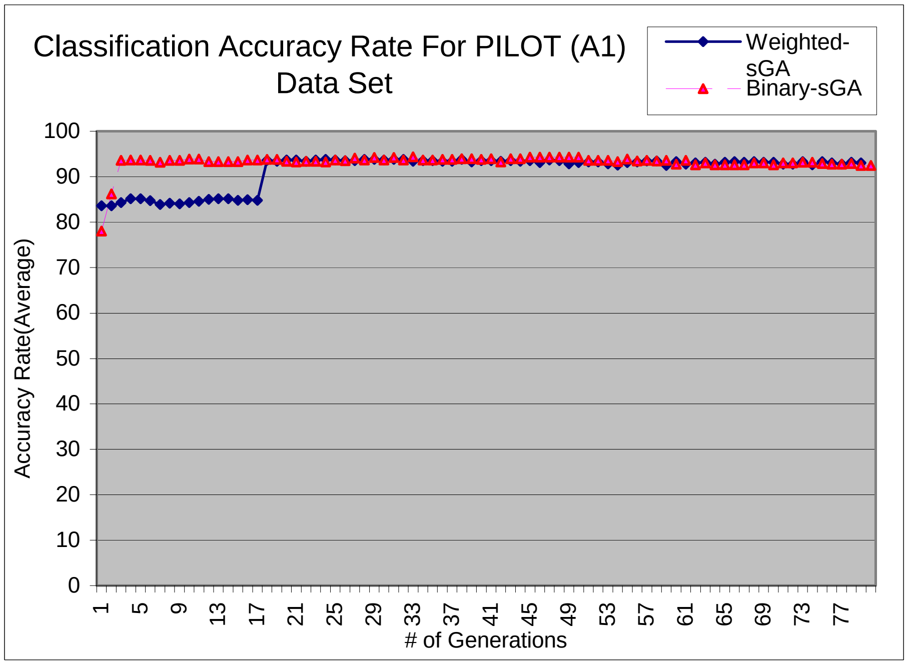


*Figure 29 Classification Accuracy Rate on Pilot (A1) Test Set For Weighted-sGA and Binary-sGA*

# *VII. ANALYSIS OF RESULTS*

## 7.1 Introduction

This chapter discusses the results of the experiments for both weighted-sGA and binary sGA. The results are analyzed according to t-variate method since the number of experiments for each dataset is only 10. Since the number of experiments is not too large, we cannot use central limit theorem and generalize our results to according to the normal distribution. We calculated the confidence intervals of each algorithm in terms of classification accuracy for each dataset. We specified two different confidence levels as 90% and 99% confidence levels during our analysis of the results.

Table 7 shows the classification accuracy of the data sets used in the experiments without feature reduction.

| Data Set | Original Feature Size | Classification Accuracy Rate Without Feature Reduction |
|---|---|---|
| Cancer | 9 | $94.84 \pm 1.16$ |
| Flag | 28 | $43.28 \pm 7.04$ |
| FLIR | 6 | $73.68 \pm 1.01$ |
| Glass | 9 | $69.71 \pm 0.63$ |
| Hepatitis | 19 | $82.75 \pm 1.89$ |
| Ionosphere | 34 | $85.40 \pm 1.85$ |
| Iris | 4 | $94.91 \pm 0.46$ |
| Liver | 6 | $60.31 \pm 3.72$ |
| Pima | 8 | $68.11 \pm 1.12$ |
| Sonar | 60 | $81.52 \pm 3.15$ |
| Soybean | 35 | $89.85 \pm 1.25$ |
| Tic-Tac-Toe | 9 | $75.30 \pm 1.52$ |
| Vote | 16 | $90.39 \pm 1.35$ |
| Wine | 13 | $73.47 \pm 0.84$ |
| Pilot (A1) | 108 | $48.95 \pm 2.07$ |
| Pilot (A2) | 108 | $59.46 \pm 1.44$ |
| Pilot (A3) | 108 | $51.78 \pm 1.80$ |
| Pilot (A4) | 108 | $59.16 \pm 3.06$ |
| Pilot (A5) | 108 | $58.32 \pm 2.08$ |

*Table 9 Classification Accuracy Rate On Real-World Data Sets Used In the Experiments Without Feature Selection and Feature Weighting*

As you can see from Table 9, when the size of the original feature set is reduced, the classification accuracy rate increases. The increase in the classification accuracy is dominant when the weighted-sGA is used. We can say that reducing the number of original features increases the overall classification accuracy rate but when the selected features are weighted we can have better classification accuracy rate.

Our experiments also show that the classification accuracy rate in test set is less than that of in training set. This is due to having fewer instances in the test set and doing classification in the training set and testing the found feature set in the test set. The bias in the training set is causing less classification accuracy rate in the test set. This is an expected result but the important point is preventing test set classification accuracy to reach a classification accuracy rate, which is not acceptable by the system user. If you look at the results on Table 8 for training set and test set, the differences of classification accuracy rate for two sets are not so significant. We can say that there is not a string bias in the training set classification process.

| Data Set | Classification Accuracy Rate (μ ± σ) for Train Set And Test Set Configurations | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Classification on Train Set | | | | | | Classification on Test Set | | | | | |
| | Weighted Classification | | | Binary Classification | | | Weighted Classification | | | Binary Classification | | |
| | (μ ± σ) | Best | Worst | (μ ± σ) | Best | Worst | (μ ± σ) | Best | Worst | (μ ± σ) | Best | Worst |
| Cancer | 97.4 ± 0.4 | 98.3 | 96.8 | 96.7 ± 0.5 | 97.7 | 96.0 | 94.5 ± 1.4 | 96.6 | 92.33 | 93.1 ± 5.0 | 97.1 | 79.4 |
| Flag | 67.9 ± 3.1 | 71.9 | 62.4 | 63.2 ± 2.8 | 68.3 | 58.7 | 46.2 ± 11.3 | 63.7 | 22.32 | 47.1 ± 6.9 | 56.8 | 34.4 |
| FLIR | 77.5 ± 1.6 | 79.9 | 75.6 | 76.2 ± 0.9 | 77.8 | 74.6 | 69.8 ± 4.1 | 76.2 | 62.64 | 67.9 ± 3.9 | 72.9 | 58.9 |
| Glass | 79.4 ± 5.4 | 92.6 | 74.6 | 73.6 ± 3.5 | 79.2 | 69.2 | 61.1 ± 5.8 | 70.2 | 51.48 | 62.9 ± 6.9 | 78.0 | 56.1 |
| Hepatitis | 94.2 ± 1.4 | 96.3 | 92.6 | 92.1 ± 1.2 | 94.4 | 90.7 | 87.7 ± 3.1 | 93.4 | 82.58 | 82.3 ± 5.1 | 89.1 | 71.7 |
| Iono | 95.7 ± 1.2 | 97.4 | 93.7 | 93.1 ± 1.1 | 94.6 | 91.3 | 85.4 ± 4.9 | 92.2 | 73.20 | 84.2 ± 3.8 | 89.4 | 78.0 |
| Iris | 96.9 ± 1.2 | 99.9 | 95.2 | 96.5 ± 1.0 | 98.0 | 95.2 | 96.8 ± 1.5 | 99.9 | 95.53 | 91.9 ± 2.6 | 95.5 | 86.6 |
| Liver | 70.5 ± 2.1 | 74.2 | 66.3 | 65.1 ± 2.7 | 69.6 | 60.5 | 58.0 ± 5.1 | 63.4 | 50.92 | 58.3 ± 7.6 | 69.2 | 47.0 |
| Pima | 75.4 ± 1.5 | 78.7 | 73.7 | 71.1 ± 1.1 | 73.1 | 69.8 | 67.3 ± 2.6 | 71.2 | 63.42 | 66.6 ± 3.5 | 70.3 | 60.8 |
| Sonar | 94.3 ± 0.9 | 95.6 | 92.8 | 91.3 ± 1.0 | 92.8 | 89.4 | 76.8 ± 7.1 | 94.8 | 67.47 | 75.8 ± 6.3 | 85.1 | 67.4 |
| Soy | 96.5 ± 0.6 | 97.4 | 95.2 | 94.7 ± 0.8 | 96.4 | 93.2 | 90.8 ± 2.3 | 93.9 | 86.66 | 86.9 ± 2.6 | 90.0 | 80.8 |
| Tic-Tac-Toe | 85.8 ± 1.1 | 87.2 | 84.5 | 81.1 ± 1.1 | 82.9 | 79.0 | 76.5 ± 1.5 | 79.7 | 74.50 | 73.4 ± 3.4 | 80.0 | 69.2 |
| Votes | 98.6 ± 0.6 | 99.9 | 97.6 | 97.8 ± 0.5 | 98.9 | 97.0 | 94.6 ± 3.8 | 99.1 | 84.58 | 94.8 ± 3.3 | 98.4 | 87.6 |
| Wine | 99.1 ± 0.4 | 99.9 | 98.3 | 97.2 ± 1.3 | 99.1 | 95.1 | 89.9 ± 3.7 | 96.1 | 84.85 | 91.8 ± 5.4 | 98.1 | 84.8 |

*Table 10 The Mean Classification Accuracy Rate and Standard Deviation For Different Data Sets*

Table 10 and 11 shows that at the 90% Confidence level there is a significant difference between weighted-sGA and binary-sGA. According to this statistics, we reject the null hypothesis defined in early chapters and accept the alternative hypothesis. We can say that weighted feature subset selection has better classification accuracy than that of binary feature subset selection.

| Data Set | Weighted-sGA Classification Confidence Interval For The Mean Accuracy | Binary-sGA Classification Confidence Interval For the Mean Accuracy |
|---|---|---|
| Cancer | (97.200075 – 97.699125 ) | (96.41279 – 97.02721) |
| Flag | (65.86412 – 69.51588) | (61.55221 – 64.86779) |
| Glass | (76.21775 – 82.58225) | (71.54067 – 75.67933) |
| Hepatitis | (93.4469 – 95.0931) | (91.44124 – 92.87876) |
| Ionosphere | (94.98226 – 96.47774) | (92.42181 – 93.77819) |
| Liver | (69.25956 – 71.74044) | (63.61916 – 66.76084) |
| Pima | (74.50575 – 76.31425) | (70.50341 – 71.83659) |
| Sonar | (93.78513 – 94.87487) | (90.71456 – 91.88544) |
| Soybean | (96.13482 – 96.86518) | (94.31628 – 95.24372) |
| Vowel | (98.73292 – 99.12708) | (97.46381 – 98.13619) |
| Votes | (98.28164 – 99.05836) | (98.11712 – 98.52228) |
| Wine | (98.80916 – 99.33084) | (96.42589 – 98. 01411) |
| FLIR | (76.57938 – 78.48062) | (75.69105 – 76.82805) |
| Iris | (96.20544 – 97.65456) | (95.95876 – 97.14124) |
| Image | (91.30469 – 92.25531) | (75.37305 – 78.16695) |
| Tic-tac-toe | (85.19239 – 86.46761) | (80.50602 – 81.87398) |

*Table 11 90% Confidence Interval, 0.95-quantile of a t-variate with 9 degrees of freedom for the Weighted-sGA and Binary-sGA Algorithms.*

| Data Set | Weighted-sGA | Binary-sGA | Mean Difference | Statistics | Value |
|---|---|---|---|---|---|
| Cancer | 97.45 | 96.72 | 0.73 | | |
| Flag | 67.69 | 63.21 | 4.48 | | |
| Glass | 79.4 | 73.61 | 5.79 | | |
| Hepatitis | 94.27 | 92.16 | 2.11 | | |
| Ionosphere | 95.73 | 93.1 | 2.63 | | |
| Liver | 70.5 | 65.19 | 5.31 | | |
| Pima | 75.41 | 71.17 | 4.24 | | |
| Sonar | 94.33 | 91.3 | 3.03 | | |
| Soybean | 96.5 | 94.78 | 1.72 | | |
| Votes | 98.67 | 97.8 | 0.87 | | |
| Vowel | 98.93 | 98.32 | 0.61 | | |
| Wine | 99.07 | 97.22 | 1.85 | | |
| Flir | 77.53 | 76.26 | 1.27 | | |
| Iris | 96.93 | 96.55 | 0.38 | | |
| Tic-tac-toe | 85.83 | 81.19 | 4.64 | | |
| | | | | Mean | 2.644 |
| | | | | Variance | 3.342 |
| | | | | Stdev | 1.828 |
| | | | | Confidence Interval ($\alpha$ = 0.1, p=0.95, t=41.761) | (1.761*(D52 /SQRT(15))) |
| | | | | Confidence interval | (1.812789 – 3.475211) |

*Table 12 Testing for a Zero Mean for Weighted-sGA and Binary-sGA at 90% Confidence level*

In Table 12, you can see the comparison of different classification algorithms and their classification accuracy performance. For some data sets GRaCCE has better classification accuracy, and for some data sets other algorithms have better performance. But to make a correct conclusion, we have to know exactly how the other algorithms are tested. The size of the training set and the size of the test set, the selected class attributes and the features, the number of experiments is all needed data to have a good statistical analysis.

| Data Set | Non-GA | | Richeldi | | DistAl | | GRaCCE | |
|---|---|---|---|---|---|---|---|---|
| | Features | Accuracy | Features | Accuracy | Features | Accuracy | Features | Accuracy |
| Cancer | 4 | 74.7 | - | - | 5.4 ± 1.4 | 99.3 ± 0.9 | 4.7±0.9 | 97.45 ± 0.43 |
| Flag | - | - | - | - | 14.0 ± 2.6 | 78.1 ± 7.8 | 9.4±2.0 | 67.69 ± 3.15 |
| Glass | 4 | 62.5 | 4 | 70.5 ± 7.8 | 5.5 ± 1.4 | 80.8 ± 5.0 | 5.4±0.9 | 79.40 ± 5.49 |
| Hepatitis | 4 | 84.6 | - | | 9.2 ± 2.3 | 97.1 ± 4.3 | 4.5±1.9 | 94.27 ± 1.42 |
| Ionosphere | - | - | - | - | 17.3 ± 3.5 | 98.6 ± 2.4 | 11.1±2.5 | 95.73 ± 1.29 |
| Liver | - | - | - | - | 4.1 ± 0.7 | 77.8 ± 4.0 | 3.7±1.42 | 70.50 ± 2.14 |
| Pima | - | - | 3 | 73.2 ± 3.8 | 3.8 ± 1.5 | 79.5 ± 3.1 | 4.3±0.95 | 75.41 ± 1.56 |
| Sonar | - | - | 16 | 76.0 ± 9.0 | 30.7 ± 3.7 | 97.2 ± 2.9 | 27.1±2.6 | 94.33 ± 0.94 |
| Soybean | - | - | - | - | 19.4 ± 2.7 | 92.8 ± 5.9 | 21.8±2.5 | 96.5 ± 0.63 |
| Votes | 4 | 97.0 | 5 | 95.7 ± 3.5 | 8.9 ± 1.8 | 98.8 ± 1.2 | 3.2±0.92 | 98.67 ± 0.67 |
| Wine | - | - | - | - | 6.7 ± 1.6 | 99.4 ± 2.1 | 5.6±1.43 | 99.07 ± 0.45 |

*Table 13 Comparison between various approaches for feature subset selection and weighting*

.

An important observation in Table 13 is that although DistAL algorithm has better classification accuracies for most of the data sets, GRaCCE selects less feature sets. This can be an important consideration when features have measurement costs. The overall cost produced by GRaCCE would be less than the neural network algorithm of DistAL.

According to Table 13, it is clear that neural network based algorithm DistAl has better performance than GRaCCE. We can say that the non-linear search capability of the neural network lead the algorithm to find very specific features that have better impact on the classification accuracy and having more features in the final subset explains why the classification accuracy rate of the DistAl algorithm is higher than sGA-based GRaCCE. Since sGA searches linearly on the feature dimensions to find the best subset of features selected from the original feature set, this linear limitation in the search process prevents GRaCCE from finding more specific features that can increase the classification accuracy rate and having less features in the final feature subset and having worse classification accuracy rate than neural network based DistAl algorithm validates this conclusion.

| Data Set | Mean Classification Accuracy Rate For Distal | Mean Classification Accuracy Rate For GRaCCE (Weighted-GA) | The Difference In the Classification Accuracy Rate of Distal and GRaCCE Algorithms |
|---|---|---|---|
| Cancer | 99.3 | 97.45 | 1.85 |
| Flag | 78.1 | 67.69 | 10.41 |
| Glass | 80.8 | 79.4 | 1.4 |
| Hepatitis | 97.1 | 94.27 | 2.83 |
| Ionosphere | 98.6 | 95.73 | 7.3 |
| Liver | 77.8 | 70.5 | 4.09 |
| Pima | 79.5 | 75.41 | 2.87 |
| Sonar | 97.2 | 94.33 | -3.7 |
| Soybean | 92.8 | 96.5 | 0.13 |
| Votes | 98.8 | 98.67 | 0.33 |
| Wine | 99.4 | 99.07 | |
| | | Sample Mean | 2.76 |
| | | Sample Variance | 13.82 |
| | | Sample Standard Deviation | 3.72 |
| | | Confidence Interval ($\alpha$ = 0.01, p=0.9995, t=4.587) | $2.76 \pm t \times 3.72/\sqrt{11}$ |
| | | 99% Confidence Interval | (-2.37896 – 7.902592) |

*Table 14 Comparison of Distal Algorithm and GRaCCE Algorithm in terms of Classification Accuracy by using 11 Different Data Sets by zero mean testing technique*

We choose the DistAL algorithm to compare or results and their results and to find if there is any significant difference between two algorithms. We compared GRaCCE and DistAL algorithm by using zero-mean-test method and the results are shown in Table 15. We use this technique because there is not more information about the experiments conducted for DisAL algorithm but since only 10 experiments are conducted for GRaCCE, t-distribution is the most suitable distribution for our experiments because of the small number of experiments.

At the 99% confidence level, since the confidence interval includes zero we can’t say that these two algorithms are statistically different.

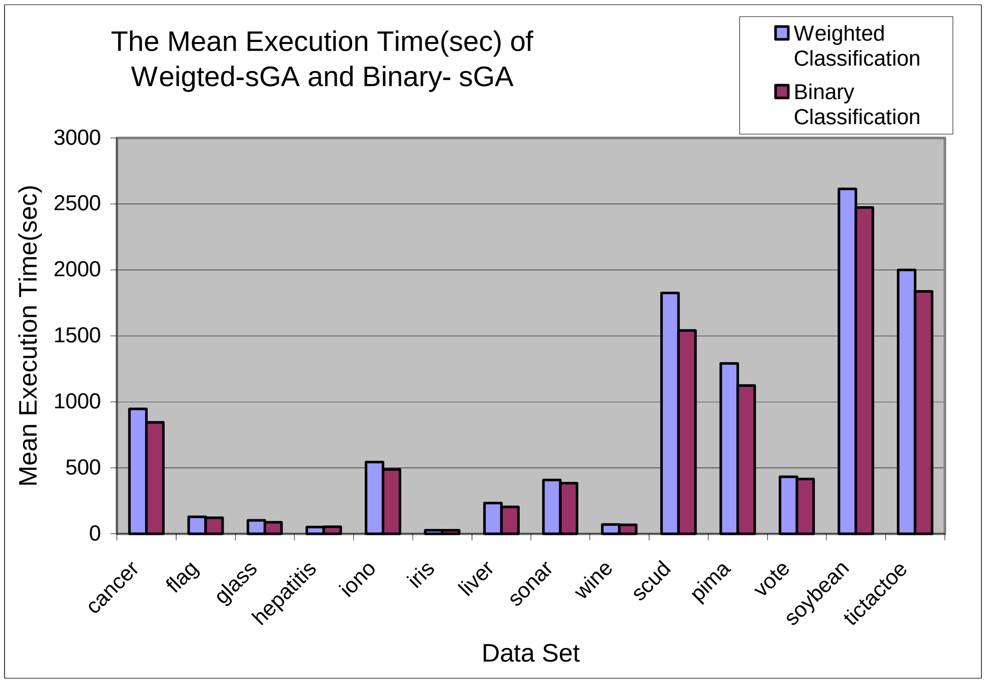


*Figure 30 The Mean Execution Time of Weighted-sGA and Binary-sGA Algorithms*

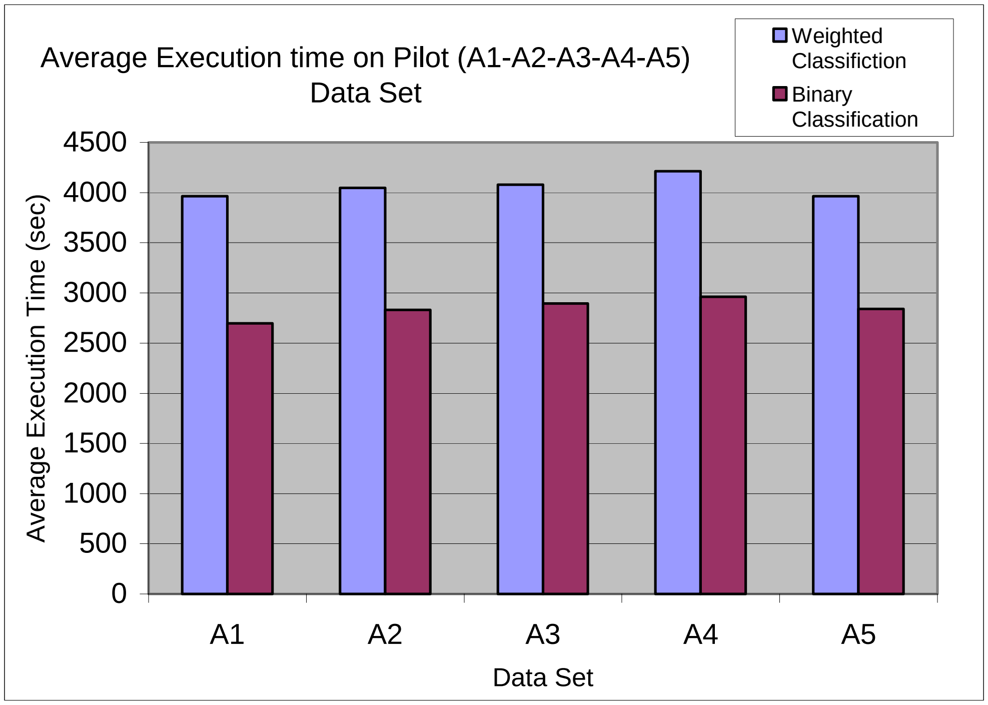


*Figure 31 The Mean Execution Time of Weighted sGA and Binary sGA Algorithms*

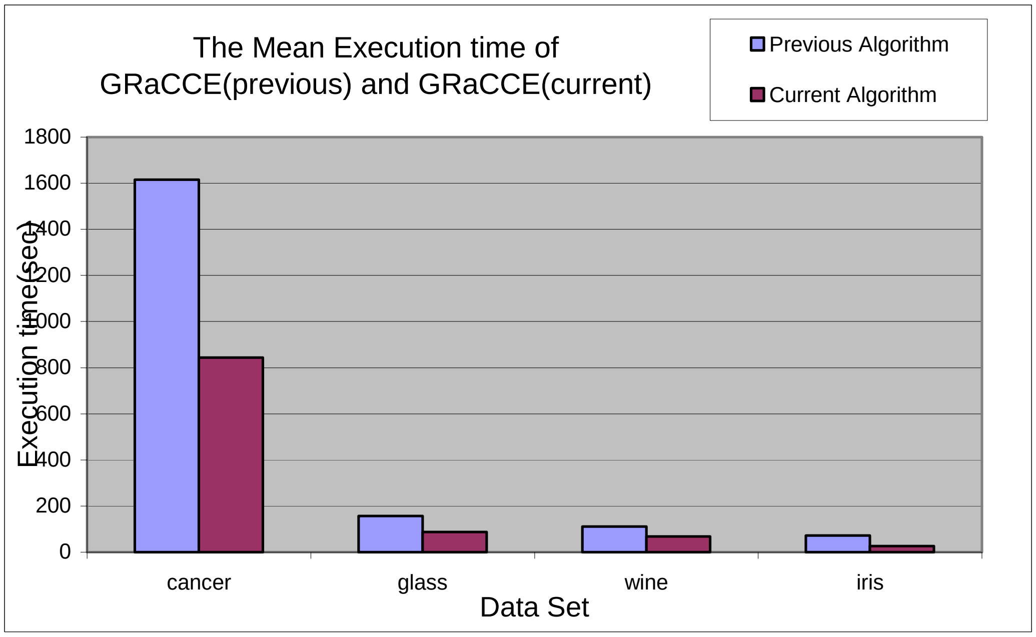


*Figure 32: The Serial Execution time of GRaCCE (previous) and GRaCCE (current)*

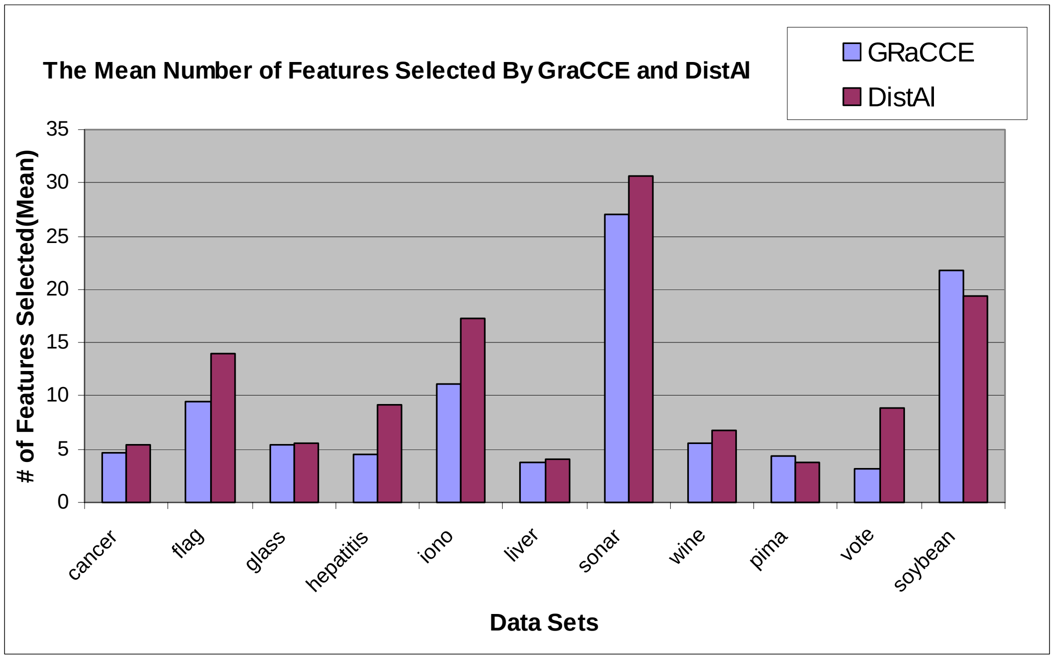


*Figure 33: Mean Number of Features Selected by GRaCCE Algorithm and DistAL Algorithm for Different Data Sets*

# *VIII. Conclusions and Future Work*

## 8.1 Conclusion

As we specified in Chapter 1, our goal was to compare the performance of the weighted-sGA and binary-sGA algorithms in terms of their classification accuracies. For that purpose we used different real-world datasets that have similar and different characteristics. We designed full factorial experiment to compare their performances. Our results show that weighting the features during the classification process lead higher classification accuracy than that of binary feature subset selection. The reason for that is weighting of features leads scales the feature dimensions in accordance with their associated weights, and this linear scaling of feature dimension makes cluster more separable than the original condition. The features that have greater weight values are highly important for the classification process and should not be ignored.

On the other hand, since the sGA algorithm uses real-value chromosomes for Weighted-Classification, the crossover and mutation operators has to be modified, because the original operators are suitable only for binary chromosomes. When real value-chromosomes are used, according to problem domain mapped to the algorithm domain, the sGA operators may produce infeasible solutions. To correct these infeasible solutions, a repair algorithm has to be implemented and integrated in to sGA algorithm used to solve the problem area. As our experiment results show in Chapter 7, these repair algorithms increase the overall execution time of the genetic algorithms. Since the sGAs are iterative algorithms and the repair algorithm has to be called for each individual after each crossover and mutation operator, the increase in the execution time of sGA is inevitable if the algorithm is run for full generation. The weighted-sGA algorithm takes almost two times longer to finish the same number of generation than binary-sGA algorithm. The solution to this problem is the development of less time consuming repair algorithms or finding another way to prevent infeasible solutions or to deal with infeasible solutions.

The results also show that GA based neural network algorithms have better statistical results when compared to the results of sGA algorithm in terms of classification accuracy, but the difference is not so significant at the 99% confidence level. The advantage of a neural network over a genetic algorithm is having the capability of non-linear search of solution areas (e.g. hyperplanes) instead of searching only on the linear dimension. But the number of features selected by neural network algorithms tends to be higher than that of GRaCCE. There should be a tradeoff between the number of selected features and the classification accuracy. When the features have measuring costs, this tradeoff can be more important, since the solution becomes a multi-objective solution which has the variables as selected number of features, classification accuracy and the cost associated with each feature.

The results also show that when the number of features increases, the number of generations for sGA to converge to a good solution increases too. The crossover operator especially causes this increase in the number of generations. In our experiments two-point crossover operator is used but when the size of chromosome is too large, the size of the chunks are also large. To increase the diversity in the population without ruining the building block, uniform crossover could be used to have more cut points in the chromosome. The preliminary experiments conducted to choose the best sGA operators that increase the diversity in the population and produce better chromosomes in the next generations show that linear ranking method increase the diversity in the population and lead slower convergence than roulette wheel selection. Probabilistic tournament selection also lead slower convergence and its convergence rate is nearly equal to the convergence rate of the linear-ranking method when the value of tournament size is equal to 2 and the probability of selecting the best individual is between 0,65 and 0.75. Increasing the tournament size and the probability of selecting the best individuals lead nearly the same convergence rate as roulette wheel selection. Our preliminary experiments also show that tournament size depends on the population size. If the population size is small, tournament size must be small too. In our case our population size is 60 and the best tournament size for this population size is decided as 2 upon many preliminary experiments with different tournament sizes. The other factor that can increase the diversity in the population is higher mutation rates.  The probability of mutation we selected is 0.1 and it introduced better chromosomes than the mutation probabilities 0.001 and 0.01. But as a side effect, higher mutation also increases the number of generation required to converge to a good solution.

In the GA, a good individual which corresponds to a nearly optimal solution of the given problem is generated by combining the schema (1) whose defining length is short, and (2 ) which has a high fitness value. In our experiments we use two-point crossover operator since it introduce better offspring and more diverse population of chromosomes than one-point crossover operator. Uniform crossover operator also introduces better offspring and more diverse population of individuals that have better fitness values than one-point crossover operator. Two-point crossover operator and uniform crossover operator have nearly same performance in terms of fitness values of the children and both introduce also diverse population. We now that if the defining length of chromosomes that have better fitness values are long, then two-point crossover operator disrupts the good building blocks more than one-point crossover operator since it selects more cut points than one-point crossover operator and after each two-point crossover operation is finished, new individuals are introduced to the next generation that have ruined building blocks. We expect from a crossover operation to cut from the edges or boundaries of the building blocks, and not to cut from the middle of building blocks since this cut destroys the good building blocks. Loosing the good building blocks causes a decrease in the total fitness value of the next generation since the next generation has worse chromosomes in terms of their fitness values than those of the previous generation because of disrupted building blocks of the individuals. Since uniform crossover operator selects more cut points than two-point crossover operator, it has more disrupting effects of the good building blocks than two-point crossover operator if the defining length of the individuals in the population is long. But our results show that since two-point and uniform crossover operator introduce better individuals that have better classification accuracy rate, the defining length of the chromosomes for feature subset weighting and feature subset selection is not long but short. Since the defining length is short, more cut points introduce more diverse individuals that have better fitness values. It is why two-point crossover operator and uniform crossover operator have better classification accuracy rate than that of one-point crossover operator.

The results for the classification accuracy rate of the different real data sets show that our sGA algorithm has promising results. Most of the time we had better results than those of non-GA based algorithms.

We showed that sGAs can be applied to feature subset selection and weighting problem easily can solve many complex classification problems in a reasonable time.

In our experiments we used only real world data sets. For feature studies, artificial data sets can be to test the GRaCCE's performance under different condition such as having databases, which do not have linearly separable clusters but non-linearly separable clusters.

## 8.2 Future Work

For classification of the unknown instances, GRaCCE only uses K-nearest neighborhood algorithm, although this algorithm produces good classification results, having almost O ($n^3$) time complexity is the biggest disadvantage for this algorithm. Future studies can try to develop faster classifier algorithms for GRaCCE environment.

The distance metric used to calculate the distances between two instances can affect the classifier accuracy, and GRaCCE uses only Euclidean distance for K-NN algorithm. New distance metrics can be developed that takes the feature correlations also into account. Euclidean distance does not take these correlations into account and it assumes that every feature dimension has an equal importance weight in the calculation of the distances.

Our research was based on only feature weighting and feature selection. We did not work on feature extraction such as by combining some features according to some linear or non-linear operations, new features can be created and added into existing feature dimension. These new features may cause less overlaps between clusters, therefore may increase the classification accuracy for k-nearest neighborhood algorithm. The feature extraction techniques can be studied in feature studies.

Although we implemented outlier removal function in GRaCCE by using z-score method, we did not use it in out experiments. The reason not to use it is because the function we developed for outlier removal assumes normal distribution for each dataset. But in real-world datasets this assumptions may not be always accurate and normal distribution may not represent the distribution of each dataset accurately. Since the removal of the outlier are very important to reduce the class overlaps, outlier removal techniques can be studied in the feature especially the which don't make assumptions about the distribution of the datasets.

Normally, there are two factors that will impact the learning of the classifier system, the distribution of classes and the distribution of attributes. During the division process of the original dataset into training set and test set, GRaCCE selects the examples from each class randomly. After the original dataset is divided into training set and test set, it is not necessary that distribution of the classes be in balance. There may not be equal number of examples from each class. But this imbalance can cause a bias toward the dominant class in the selected examples and can increase the classification error rate or prevent the resulting feature subset from being generalized to future unknown examples. Consequently, it is necessary to introduce a new division method of original dataset into training set and test set to prevent the skewed class distribution.

In our experiments we did not used the "n-fold cross-validation" technique where the data set is divided into n equal part and each time n-1 different part is taken as training set and the other one part as test set and the classification accuracy is calculated for all combinations of training sets and test sets.  In future studies, this technique can be used to increase the reliability of the KNN classifier system.

## *A. Genetic Rule and Classifier Construction Environment (GRaCCE)*

The algorithm employed by GRaCCE has five distinct phases: feature selection, partition generation, data set approximation, region identification, and region refinement. Each of these pahses is discussed in the sections that follow. It is worth noting that parameters described in these discussions are user-tunable within the GRaCCE framework.

### A.1 Feature Selection Phase

In this first phase of the algorithm, an optional feature subset selection process is performed to reduce the dataset's dimensionality. Feature subset selection is accomplished by using the GA-based approach developed by Sklansky and Siedlecki [35]. In their technique, each database feature is assigned to a gene in the GA's chromosome structure. Each gene's binary allele determines if the feature is ignored (value=0) or utilized (value=1). The GA's objective function evaluates each individual's fitness based on a K Nearest Neighbor (KNN) Algorithm [36, pages 55-57]; the fittest individuals are those that achieve the best accuracy with the fewest enabled features. When the GA terminates, the data set's dimensionality is reduced b eliminating those features disabled in the fittest individual. Eliminating unnecessary features helps overcome the curse of dimensionality [36, page 7] during training and also simplifies the structure of the decision rules [19].

### A.2  Partition Generation Phase

The purpose of this phase is to identify all potential class boundaries. The primary obstacle to generating cluster partitions is finding good boundary points when classes are not linearly separable. An elegant technique for solving this difficult problem was developed by Lee and Landgrebe [37]. According to their technique, class seperation can be enhanced in the training data by removing points misclassified by KNN algorithm. This winnowing process is iterated until none of the remaining points are misclassified. After completing this procedure we are left with the reduced data set. Because separation between classes is improved, the points that border neighboring classes are now easier to identify [19].

These boundary points are now used to estimate the class boundaries. If we assume a Gaussian distribution for each class, the decision boundary h(X) separating any two classes ($\omega_i$, $\omega_j$) in the d dimensional feature space is given by Equation 1. The equations needed to compute the anchor point $X_0$ through which h(X) runs can also be found [37]. For each boundary point, mean ($\hat{\mu}$) and covariance matrix ($\hat{\sum}$) for each cluster are estimated from its neighboring points of the same cluster. As a result, each generated partition is a local estimate how to best separate $\omega_i$ and $\omega_j$.

$$h(X) = 0.5(X - \hat{\mu}_i)^t \hat{\sum}_i^{-1}(X - \hat{\mu}_i) - 0.5(X - \hat{\mu}_j)^t \hat{\sum}_j^{-1}(X - \hat{\mu}_j) + 0.5 \ln \frac{\left|\hat{\sum}_i\right|}{\left|\hat{\sum}_j\right|} \qquad \text{(A.2.1)}$$

## A.3 Data Set Approximation Phase

Any complete set of CH regions must encompass the boundary points described. Then, training data set can be replaced with the set of boundary points fro the duration of the search [19]. This substitution has the advantage of accelerating the search by reducing the number of comparisons needed to determine the fitness. Then all boundary points are grouped by class. For each class ($\omega_i$), a weight (ρ) is computed for each boundary point equal to the percentage of $\omega_i$ training points that are closest to it (relative to all other boundary points in the same class). This weight represents the relative density of data around a boundary point belonging to $\omega_i$ . Once weights are computed for all N boundary points in $\omega_i$ , weights are sorted in decreasing order for their assigned class [19]. The next section shows how these weights enable GRaCCE to better assess the fitness of a CH region containing a set of boundary points.

## A.4 Region Identification Phase

GRaCCE now conducts a series of searches (by class) to identify the CH regions sought. This is accomplished by finding a group of partitions that isolate boundary points belonging to the sane class with a user specified degree of purity ($\gamma_{user}$). As in the feature selection phase, GRaCCE uses a GA to accomplish the search. If GRaCCE generates l partitions from a dataset consisting of m classes, the GA's chromosome structure (I) is encoded such that each gene corresponds to a specific partition; the binary allele of the gene shows the partition's status (0 if disabled, 1 if enabled). Thus the region defined by each individual ( $\overset{\rightarrow}{a}$ a $\in$ I) in the GA population (P) is determined by which of its partitions in $\overset{\rightarrow}{a}$ are enabled [19].

## A.5 Region Refinement Phase

The goal of this phase is to improve the regions found in the previous phase. This is accomplished in three ways. First, GRaCCE eliminate regions that contain fewer than a specified percentage (nominally two) of their target class data. Since small regions tend to use a large number of partitions, removing them helps avoid over training while reducing the unnecessary complexity. Second, each boundary enabled in $\overset{\rightarrow}{a}$ is tested to ensure it contributes to fitness function $\Phi(\overset{\rightarrow}{a})$; any boundary found to be extraneous is removed. At this point, it is considered the regions to be fully defined [19]. GRaCCE now recomputes the covariance matrix of each region based on the data points assigned to it. With the updated covariance matrix, the Equation 1 is used to adjust the orientation of each partition reflect the boundaries of the regions they separate; only those adjustments that improve the classification accuracy of the training set are kept. At the en of this phase, GRaCCE presents the simplified version of the regions and their respective decision boundaries that were derived in the previous section. Because these decision boundaries separate each region from the rest of the data space, we can think of each region as a rule for classifying the data and each of its partitions as conditions for that rule.

## B. Fitness Landscapes

Binary-sGA and Weighted-sGA Fitness Landscapes for Different Real-World Data Sets

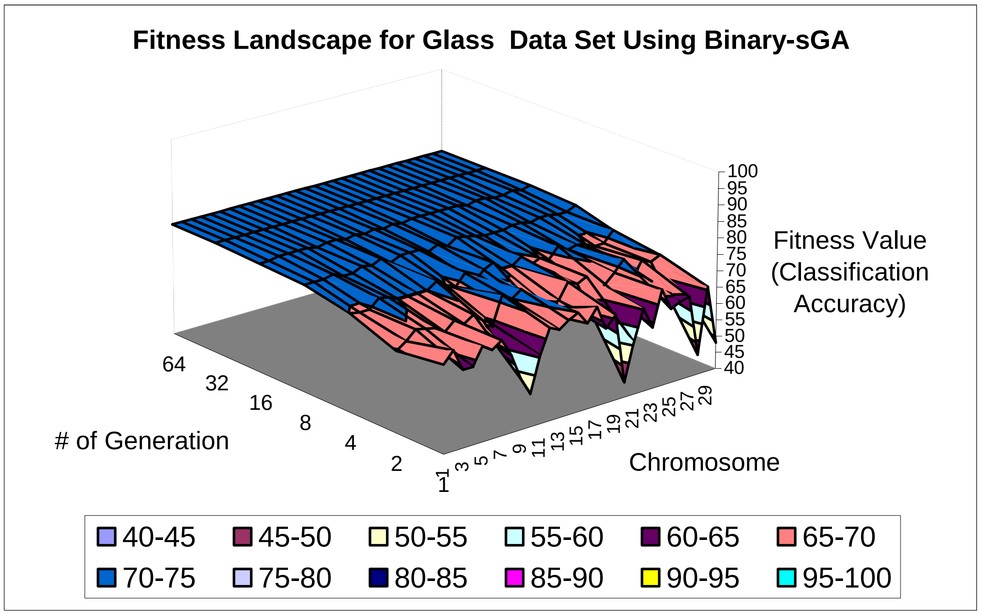


*Figure 34 Fitness Landscape for Glass Data Set Using Binary-Sga*

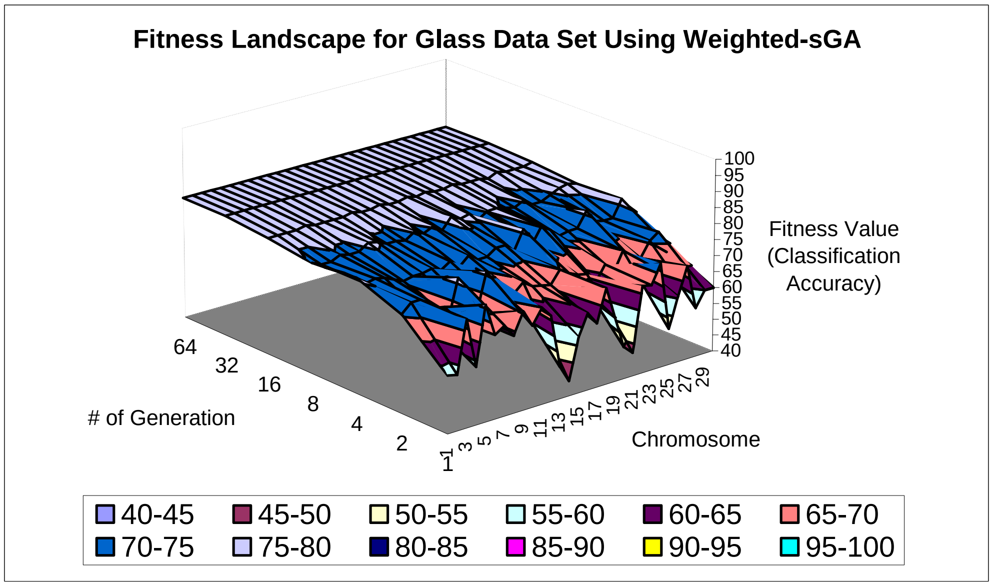


*Figure 35 Fitness Landscape for Glass Data Set Using Weighted-sGA*

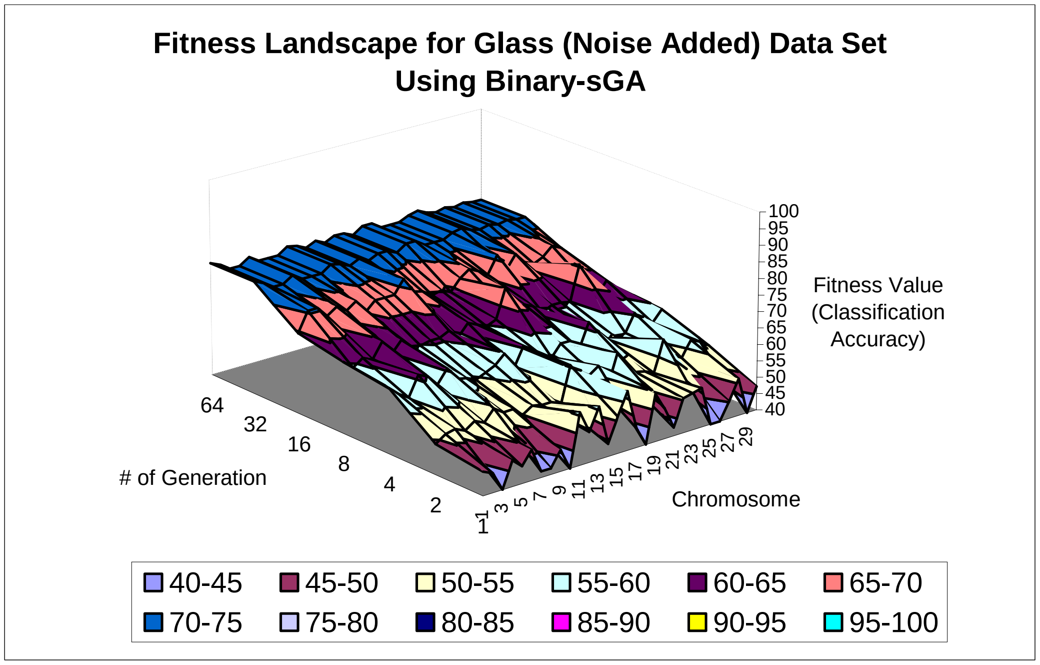


*Figure 36 Fitness Landscape for Glass (Noise Added) Data Set Using Binary-sGA*

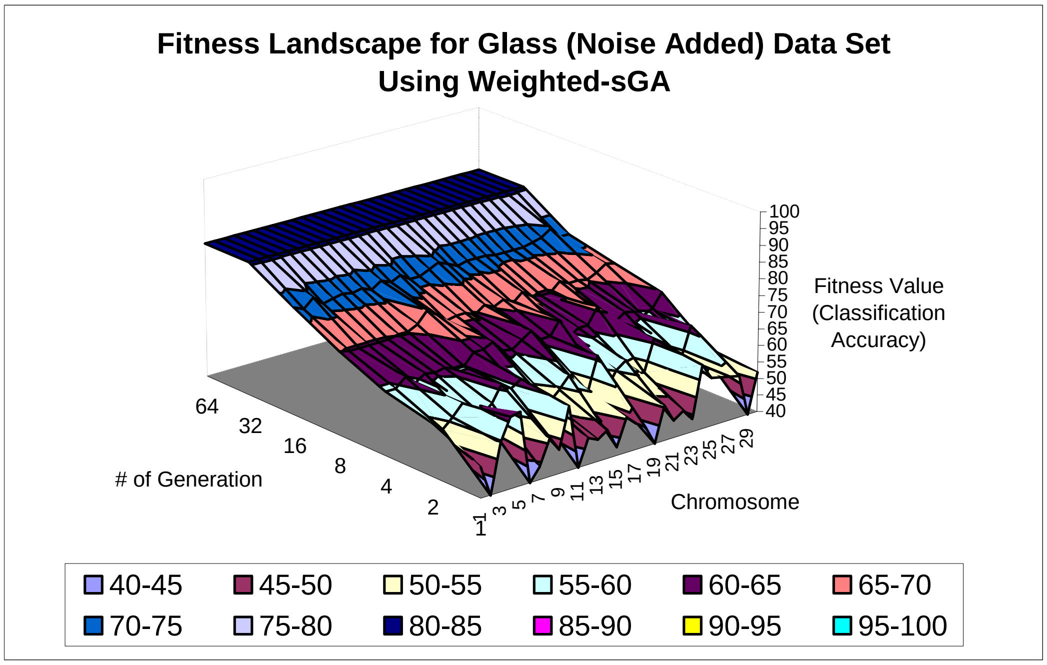


*Figure 37 Fitness Landscape for Glass (Noise Added) Data Set Using Weighted-sGA*

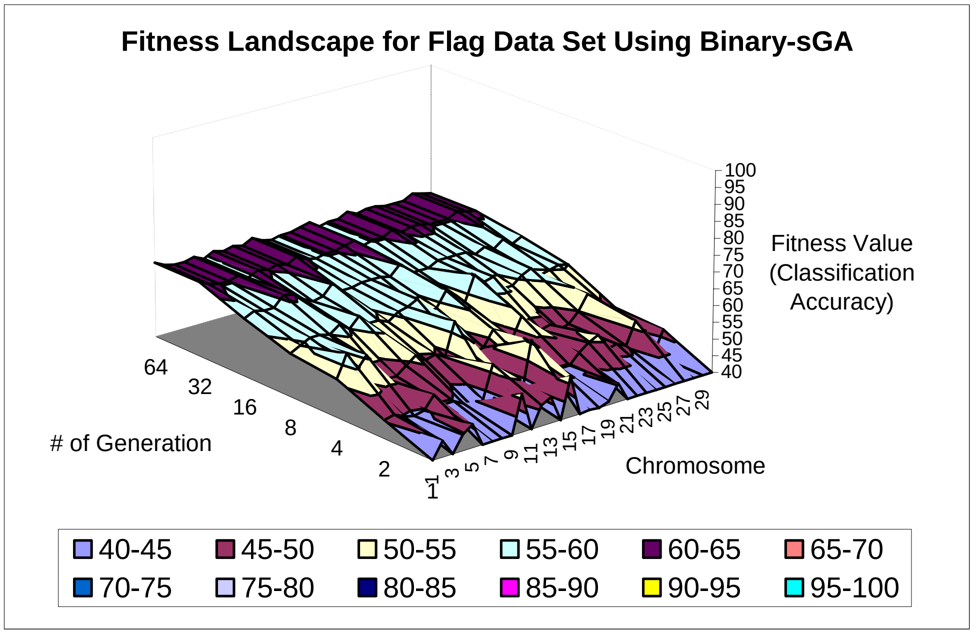


*Figure 38 Fitness Landscape for Flag Data Set Using Binary-sGA*

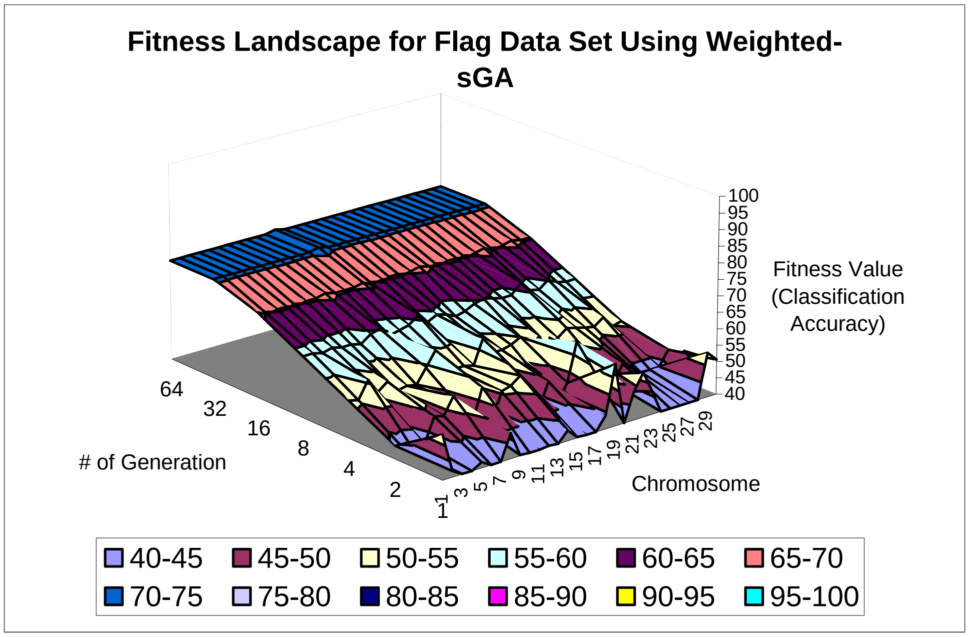


*Figure 39 Fitness Landscape for Flag Data Set Using Weighted-sGA*

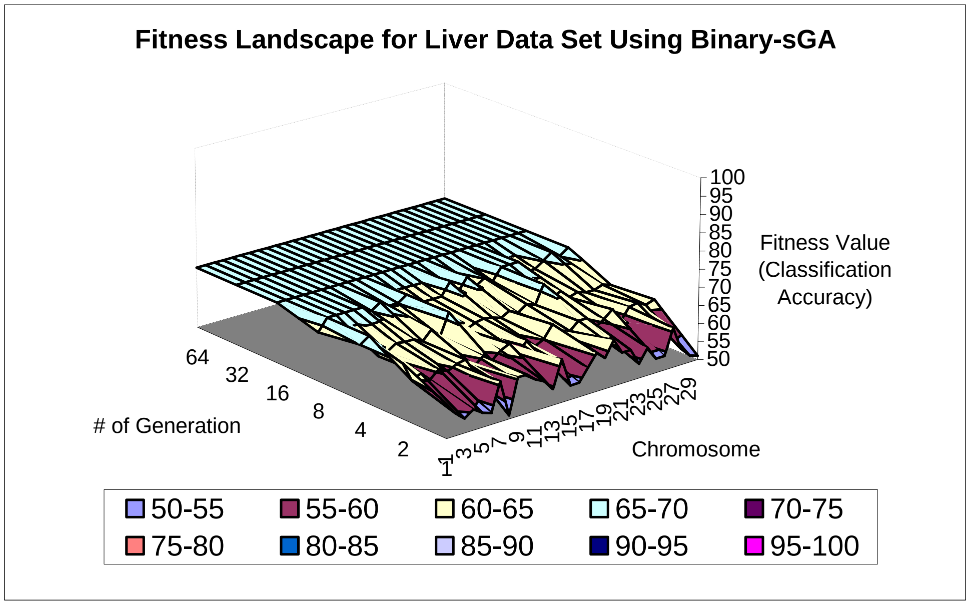


*Figure 40 Fitness Landscape for Liver Data Set Using Binary-sGA*

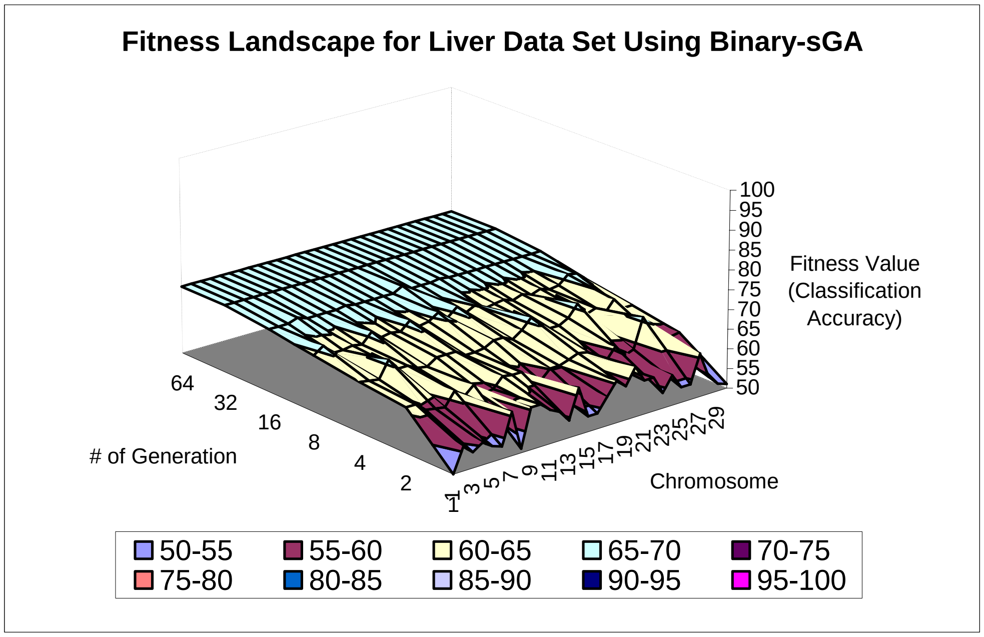


*Figure 41 Fitness Landscape for Liver Data Set Using Weighted sGA*

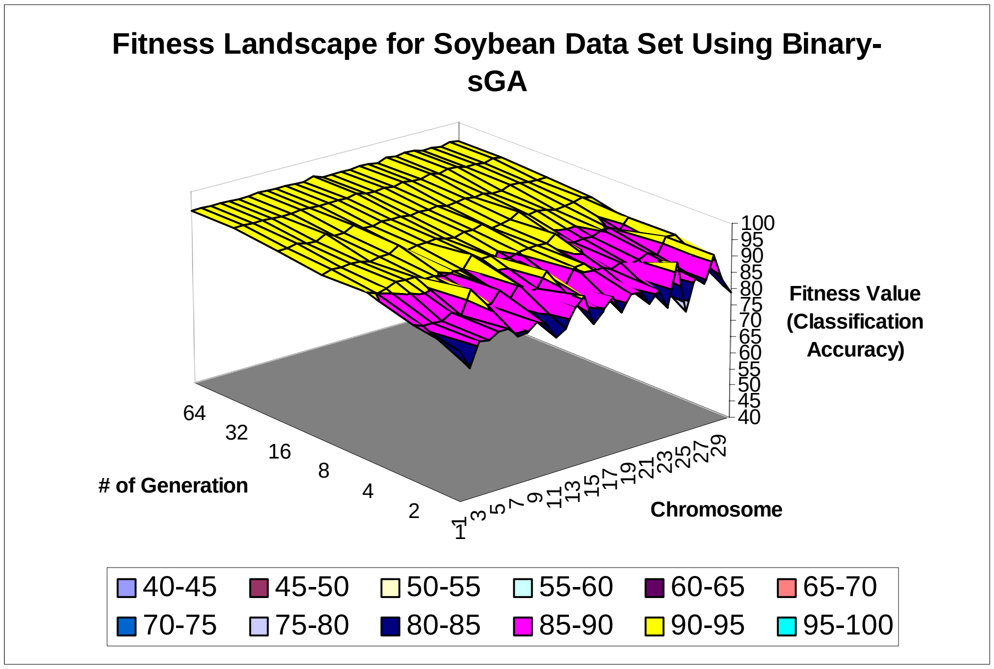


*Figure 42 Fitness Landscape for Soybean Data Set Using Binary-sGA*

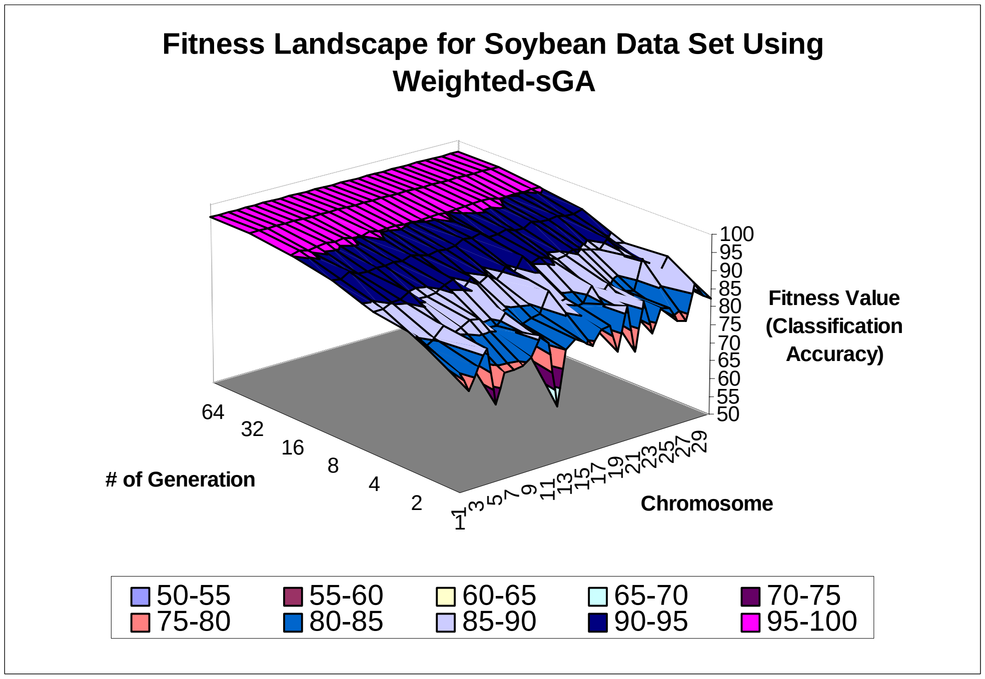


*Figure 43 Fitness Landscape for Soybean Data Set Using Weighted-sGA*

## *C. Extended Results Of The Experiments for Real-World Datasets*

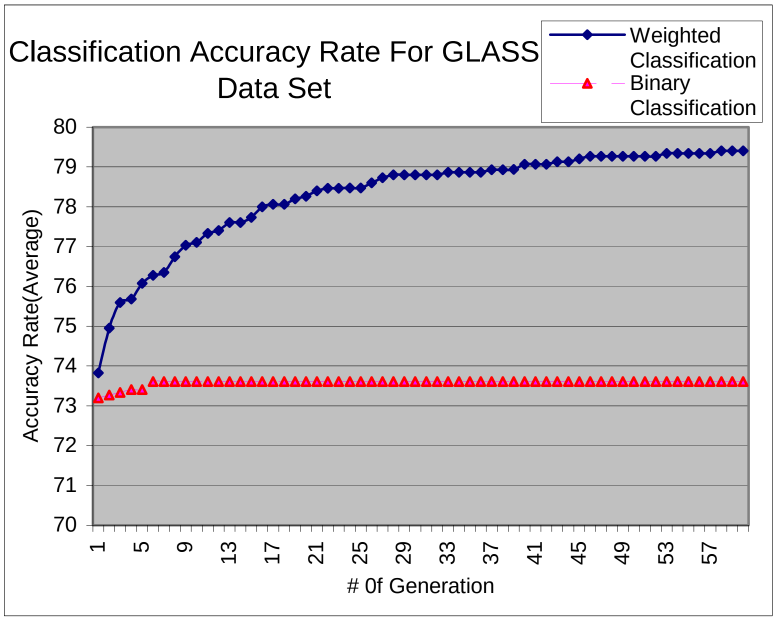


*Figure 44 Classification Accuracy Rate on Glass Train Set For Weighted-sGA and Binary-sGA*

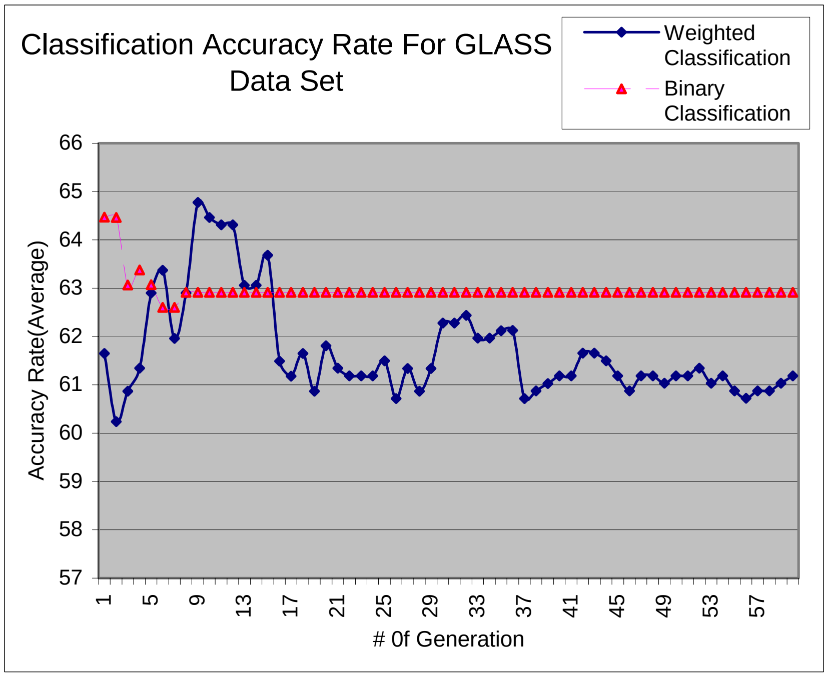


*Figure 45 Classification Accuracy Rate on Glass Test Set For Weighted-sGA and Binary-sGA*

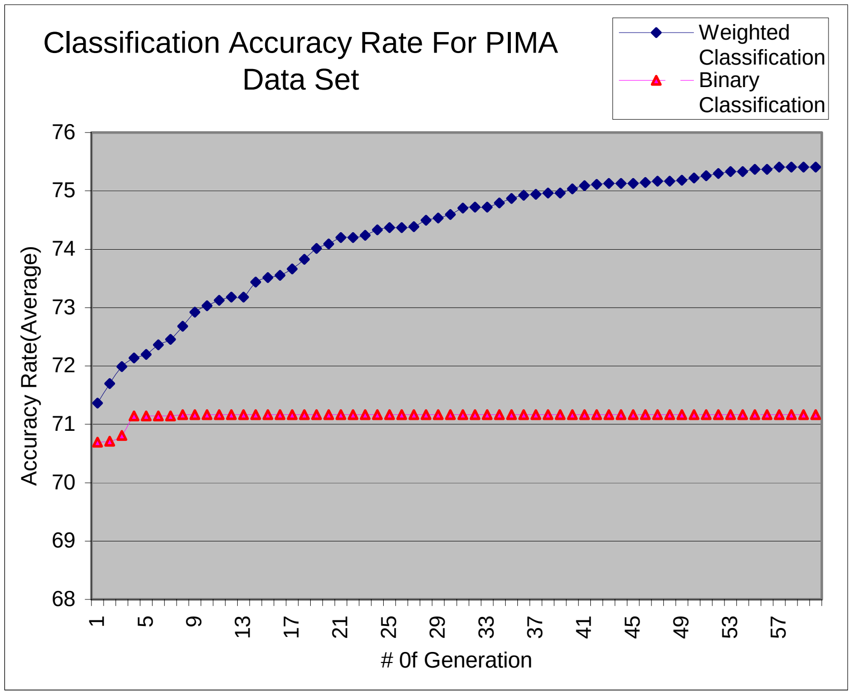


*Figure 46 Classification Accuracy Rate on Pima Train Set For Weighted-sGA and Binary-sGA*

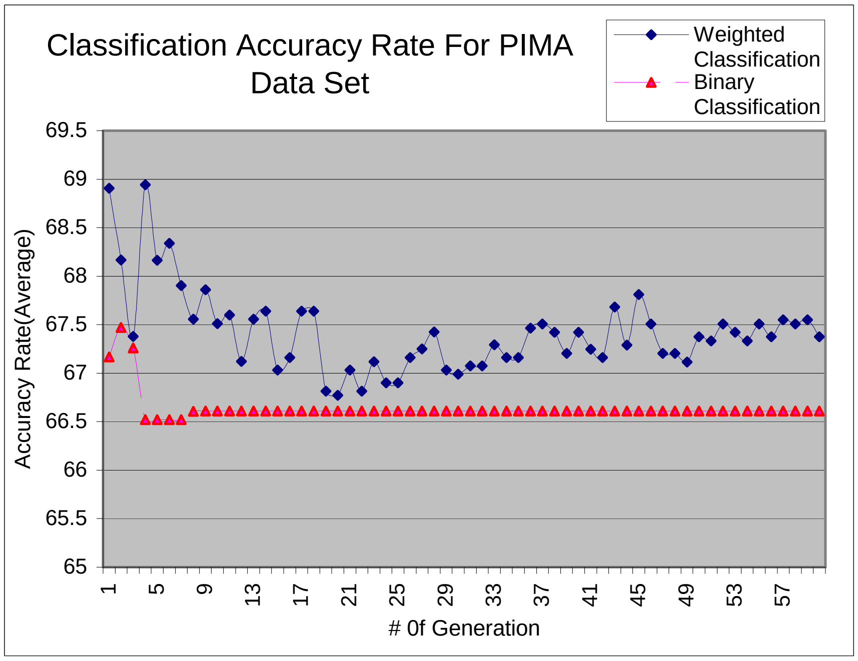


*Figure 47 Classification Accuracy Rate on Pima Test Set For Weighted-sGA and Binary-sGA*

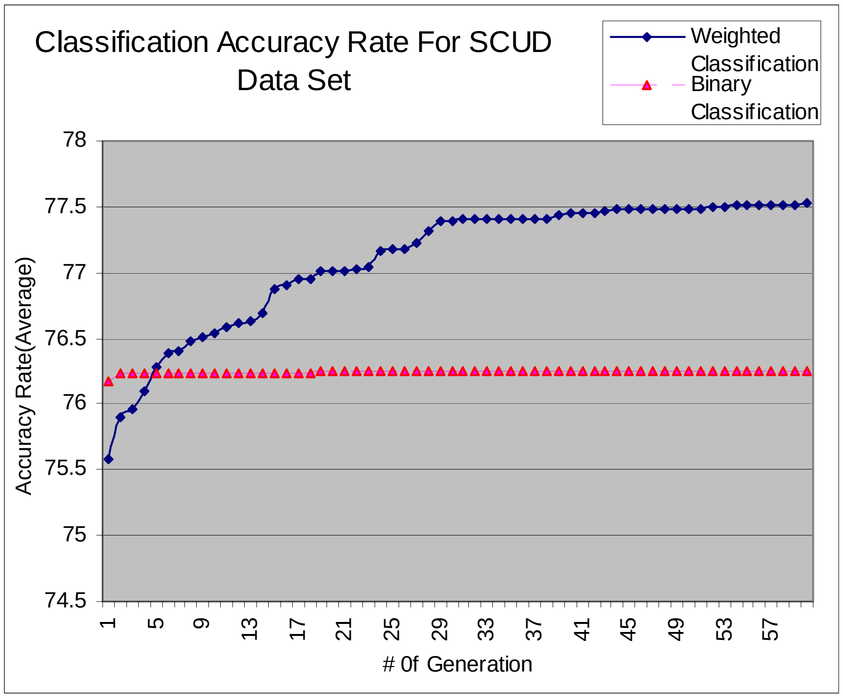


*Figure 48 Classification Accuracy Rate on FLIR Train Set For Weighted-sGA and Binary-sGA*

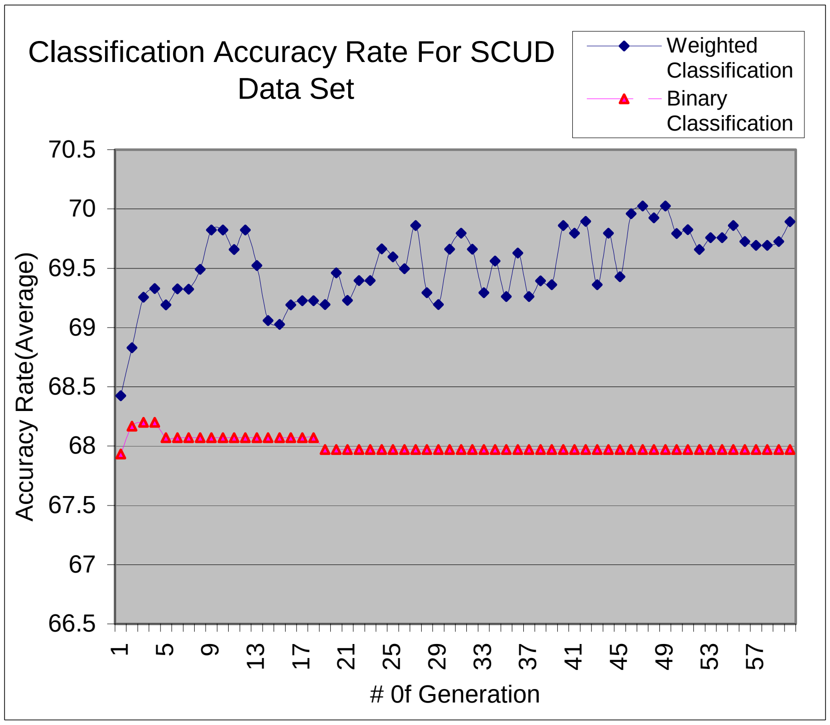


*Figure 49 Classification Accuracy Rate on FLIR Test Set For Weighted-sGA and Binary-sGA*

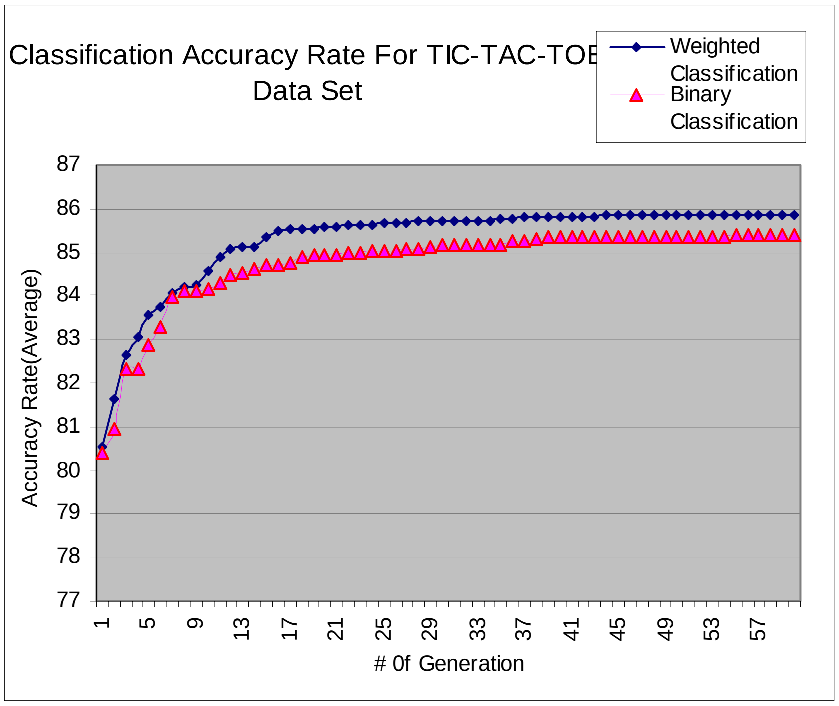


*Figure 50 Classification Accuracy Rate on TicTacToe Train Set For Weighted-sGA and Binary-sGA*

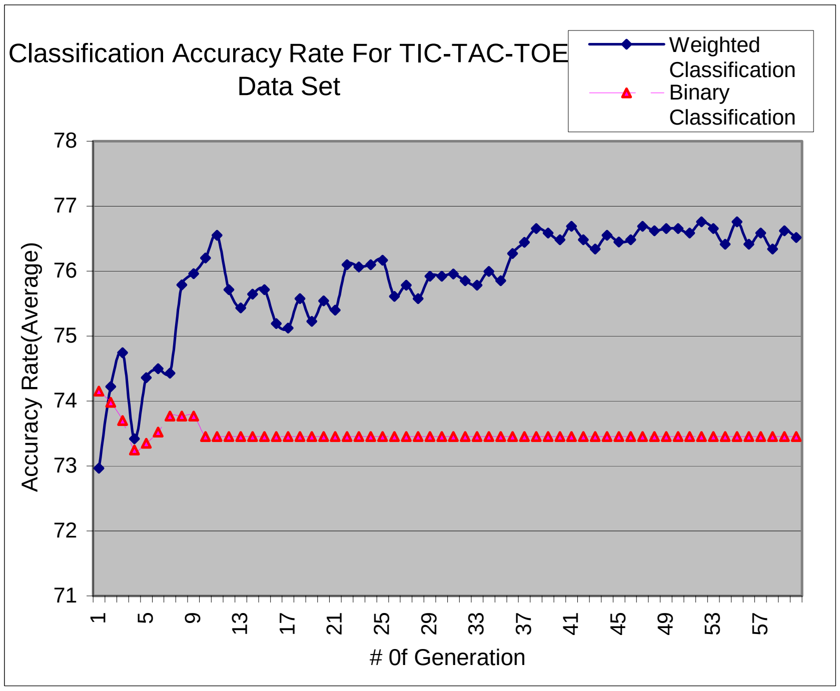


*Figure 51 Classification Accuracy Rate on Tic-Tac-Toe Test For Weighted-sGA and Binary-sGA*

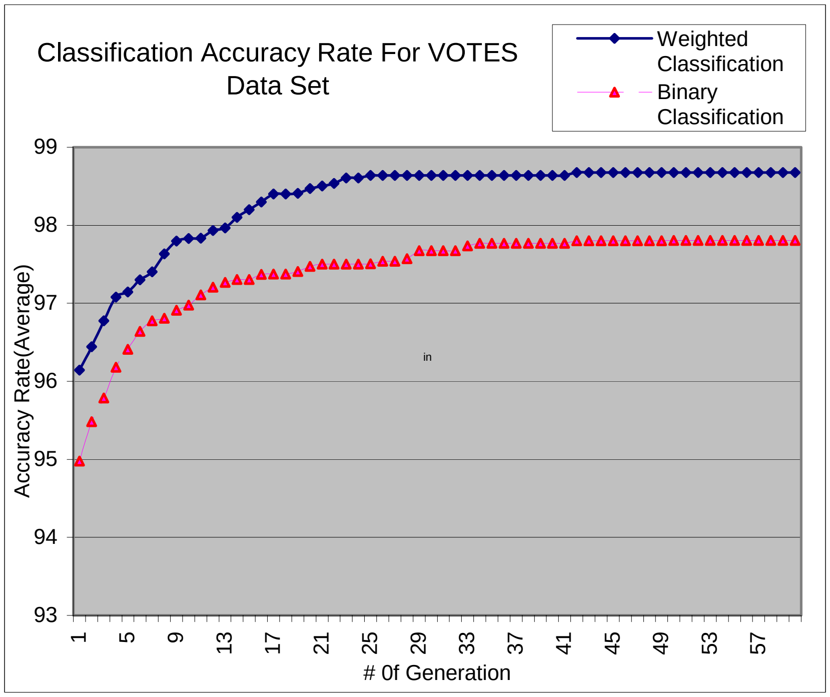


*Figure 52 Classification Accuracy Rate on Votes Train For Weighted-sGA and Binary-sGA*

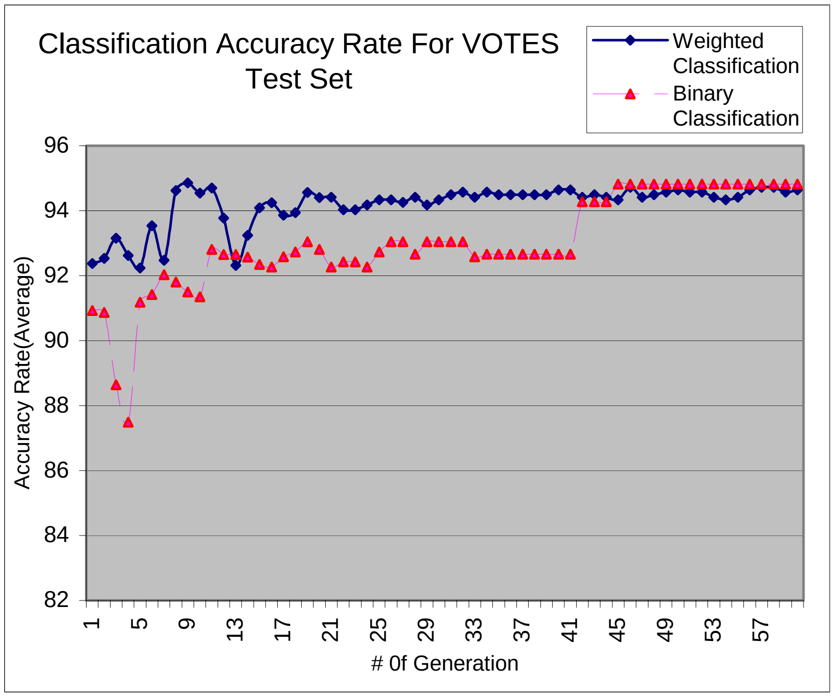


*Figure 53 Classification Accuracy Rate on Votes Test Set For Weighted-sGA and Binary-sGA*

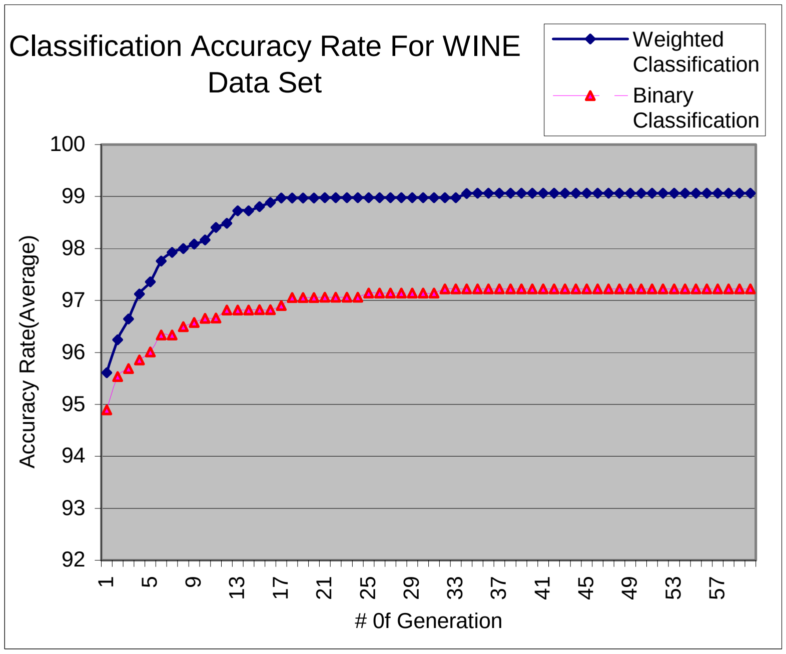


*Figure 54 Classification Accuracy Rate on Wine Train Set For Weighted-sGA and Binary-sGA*

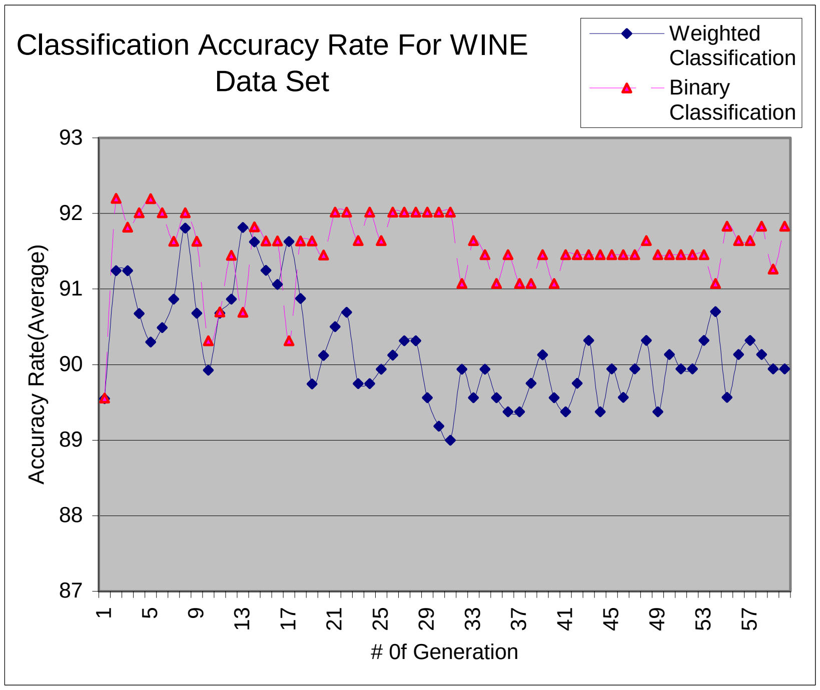


*Figure 55 Classification Accuracy Rate on Wine Test Set For Weighted-sGA and Binary-sGA*

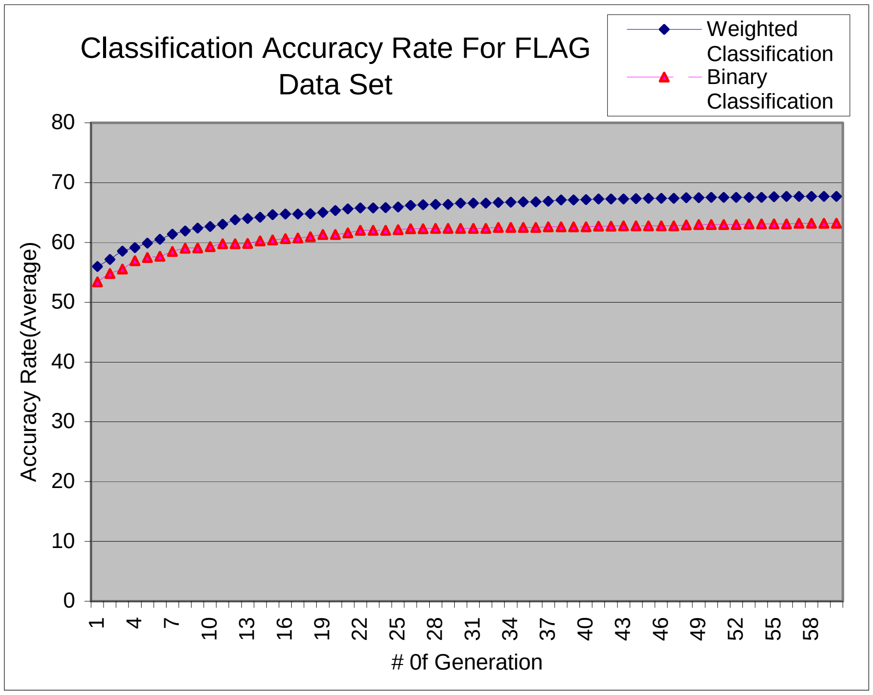


*Figure 56 Classification Accuracy Rate on Flag Train Set For Weighted-sGA and binary-sGA*

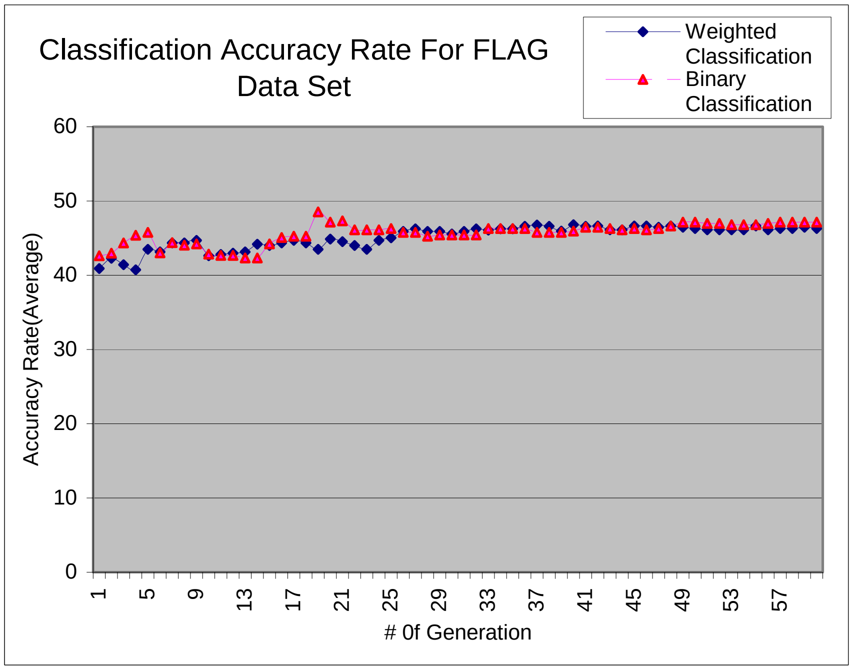


*Figure 57 Classification Accuracy Rate on Flag Test Set For Weighted-sGA and Binary-sGA*

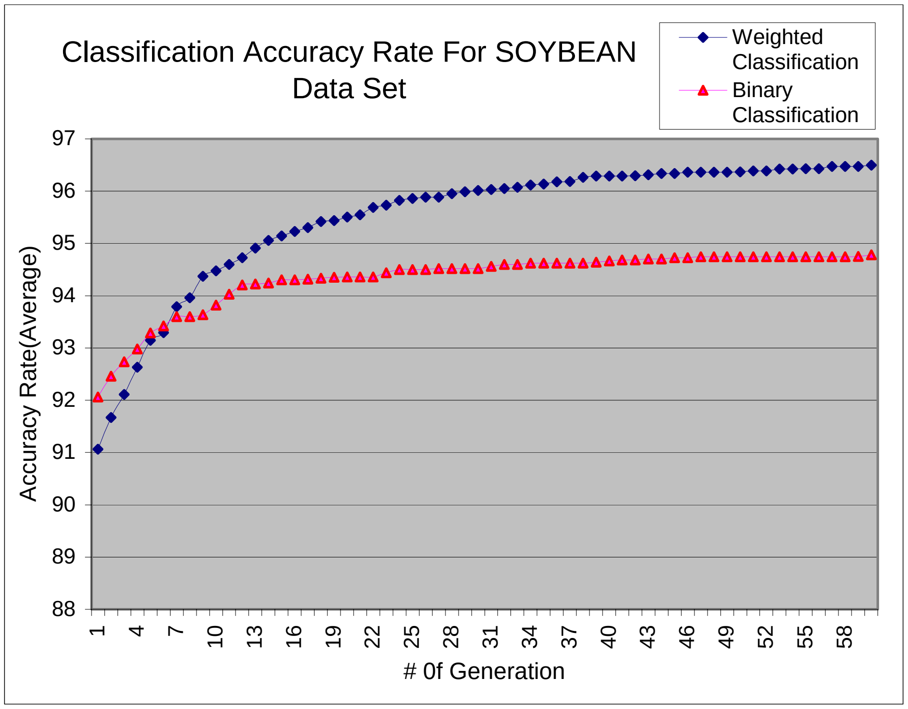


*Figure 58 Classification Accuracy Rate on Soybean Train For Weighted-sGA and Binary-sGA*

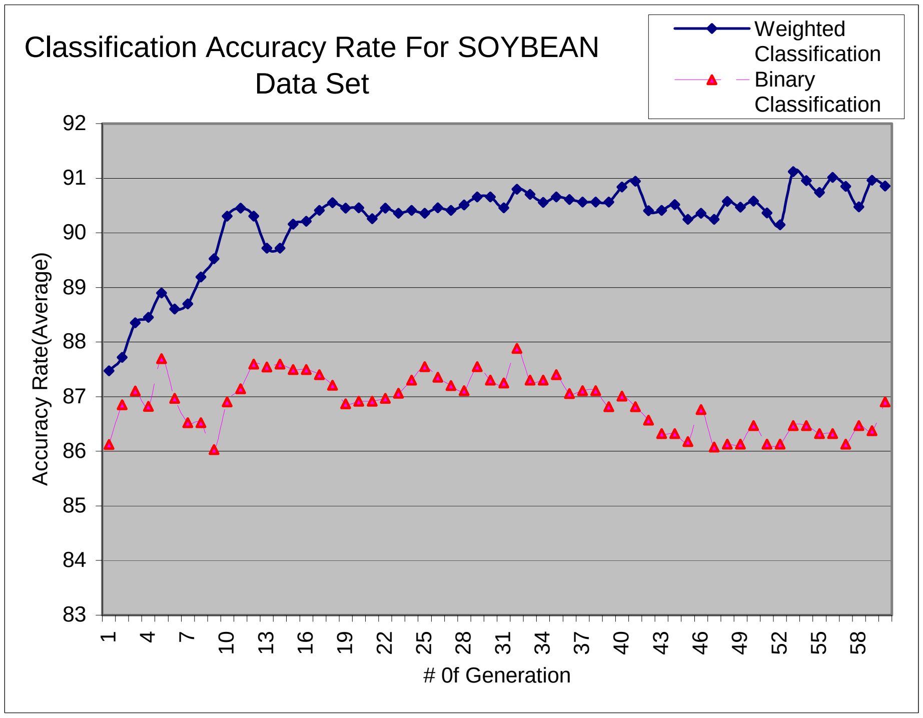


*Figure 59 Classification Accuracy Rate on Soybean Test For Weighted-sGA and Binary-sGA*

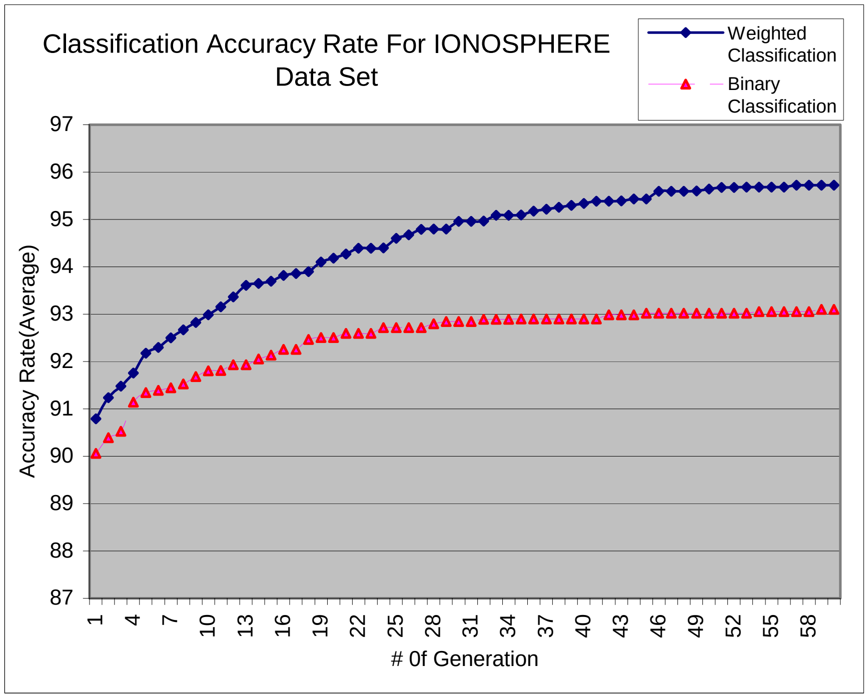


*Figure 60 Classification Accuracy Rate on Ionosphere Train Set For Weighted-sGA and Binary-sGA*

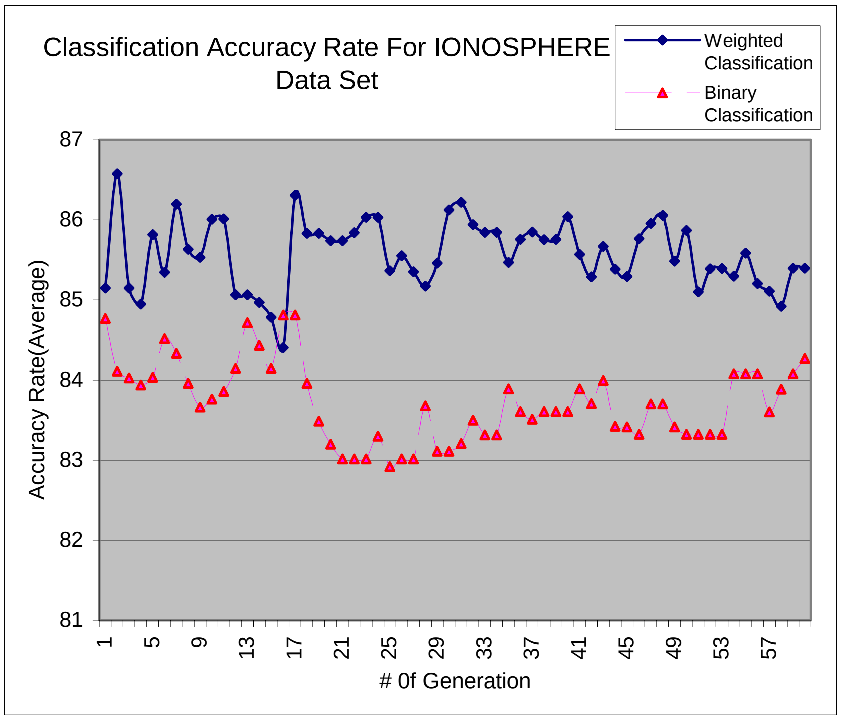


*Figure 61 Classification Accuracy Rate on Ionosphere Test Set For Weighted-sGA and binary-sGA*

## *D. Multi-objective optimization*

For many real world decision-making problems there is a need for simultaneous optimization of multiple objectives. [30]

In feature subset selection and weighting problem one of the objectives is to increase the classification accuracy of the classifier systems. If this is the only objective then it is said to be a single-objective optimization. But usually we want minimum number of features selected to classify an unknown object and at that time we want the highest classification accuracy with minimum number of selected features. Obviously there are two objectives in this optimization problem, reducing the number of features and increasing the classification accuracy. But it doesn’t necessarily mean that reducing the number of features always increase the classification accuracy. During the search process of the genetic algorithms, GA finds larger feature subsets that have better classification accuracy. Normally if equal weights are given to these two objectives, then the algorithm either tries to find the minimum number of features or maximum classification accuracy. But the solution can be a feature subset, which has zero features. But this is not a feasible solution since we need at least on feature to represent a real-world object. If the features have measurement costs then we may want to reduce the total cost of the selected feature subset while reducing the number of selected features and increasing the classification accuracy. But any changes to the value of any objective may have an opposite effect on the other objectives. For example increasing the classification accuracy may increase the total cost of the selected feature subset. So, there must be a tradeoff among each objective. Clearly this is a multi-objective problem having three objectives.

Such multi-objective optimization problems require separate techniques, which are very different to the standard optimization techniques for single objective optimization. It is very clear that of there are two objectives to be optimized, it might be possible to find a solution, which is best with respect to the first objective, and another solution, which is the best with respect to the second objective. [30]

It is convenient to classify all potential solutions to multi-objective optimization problem into dominated solutions and non-dominated (Pareto-optimal) solutions. As solution s is dominated if there exits a feasible solution y not worse than on all coordinates, i.e., for all objectives $f_i$ (I=1, . . . , k):

$f_i(x) \leq f_i(y)$ for all $1 \leq i \leq k$. [1]

If a solution is not dominated by any other feasible solution, we call it non-dominated (or Pareto-optimal) solution. All Pareto-optimal solutions might be of some interests; ideally, the system should report back the set of all Pareto-optimal points.

There are some classical methods for multi-objective optimization [31]. These include a method of objective weighting, where multiple objective functions $f_i$ are combined into one overall objective function F:

$$\mathrm{F}(\mathrm{x}) = \sum_{i=1}^{k} w_i \, f_i(x) \qquad \text{(C.1)}$$

Where the weights $w_i$ э [0..1] and $\sum_{i=1}^{k} w_i = 1$. Different weight vectors provide different Pareto-optimal solutions. Another method (method of distance functions) combines multiple objective functions into one on the basis of demand level vector y:

$$F(x) = \left(\sum_{i=1}^{k} | f_i(x) - y_i |^r\right)^{\frac{1}{2}} \qquad \text{(C.2)}$$

Where (usually) r = 2 (Euclidean metric). [30]

Multi-objective optimization enjoyed some interest in the GA community. In 1984 Schaffer [32] developed VEGA program (for Vector Evaluated Genetic Algorithm), which was an extension of the GENESIS

[1] For maximization problems, otherwise the less equal inequality should be replaced by greater equality.

program [33] to include multi-criteria functions. The main idea behind the VEGA system was a division of the population into (equal sized) subpopulations; each subpopulation was “responsible” for a single objective. The selection procedure was performed independently for each objective, but crossover was performed across subpopulation boundaries. Additional heuristics were developed (e.g., wealth redistribution scheme, crossbreeding plan) and studied to decrease a tendency of the system to converge towards individuals, which were not the best with respect to any objective.

## *E. Random number generators*

The subject of random number generation has a very extensive literature (Knuth 1969), so this section provides only a brief introduction to the topic. In fact, most programming language libraries include at least one random number generator, so it may be not necessary to implement one as part of an evolutionary algorithm. However, it is important to be aware of the properties of the random number generator being used, and to avoid the use of poorly designed generator. For example, Booker (1987) points out that care must be taken when using simple multiplicative random number generators to initialize a population, because the values that are generated may not be randomly distributed in more than one dimension. Booker recommends generating random populations as usual, and then performing repeated crossover operation with uniform random pairing. Ideally, this would be done to the point of stochastic equilibrium, meaning that the probability of occurrence of every schema is equal to the product of the proportions of it defining length [29].

One commonly used method of generating a pseudorandom sequence is the linear congruential method, defined by the relation

$X_{n+1} = (a_r X + c_r) \bmod m_r \; n \geq 0$

Where $m_r > 0$ is the modulus, X0 is the starting value (or seed), and ar and cr are constants in the range $[0, m_r)$. The properties of the linear congruential method depend on the choice made for the constants $a_r$, $c_r$, and $m_r$. (See book by Knuth (1969) for a thorough discussion). Reasonably good results can be obtained with the following values on a 32-bit computer (Press et al 1998):

$a_r = 4096$

$c_r = 150889$

$m_r = 714025$

The following routine uses the linear congruential method to generate a uniformly distributed random number in the range [0, 1):

Input: the current random seed, Seed

Output: $u_r$, a uniformly distributed random number in the range [0,1);

The seed is updates as a side effect

```
1    Urand (Seed)
2    Seed $\leftarrow$ ($a_r$Seed + $c_r$) mod $m_r$;
3    $u_r \leftarrow$ Seed / $m_r$;
4    return $u_r$;
```

Given Urand above, the following generates a uniformly distributed real value in the range [a, b):

Input: lower bound a, upper bound b.

Output: u, a uniformly distributed random number in the range [a,b).

```
1    U (a, b)
2    ur$\leftarrow$Urand (Seed);
3    u$\leftarrow$a + (b-a) $u_r$;
4    return u;
```

Note that the above procedure always returns a value strictly less than the upper bound b. The following generates an integer value from the range [a, b] (inclusive of both endpoints):

Input: lower bound a (an integer), upper bound b (an integer).

Output: I, a uniformly distributed random integer in the range [a, b].

```
Irand (a,b):
ur←Urand(Seed);
i←a + $\lfloor (b-a+1)\, u_r \rfloor$;
return i;
```

Finally, evolutionary algorithms often require the generation of a randomized shuffle of permutation of objects. For example, it may be desired to shuffle a population before performing pair wise crossover. The following code implements a random shuffle:

Input: perm, an integer array of size n.

Output: perm, an array containing a random permutation of values from 1 to n

```
Shuffle(perm):
for i←1 to n do
permit[i]←i;
od
for i←1 to n-1 do
        j←Irand(i, n);
{swap items i and j}
temp←perm[i]
perm[i]←perm[j]
perm[j]←temp;
od
return perm;
```

# *Bibliography*

1 Jiawei Han and Micheline Kamber, Data Mining Concepts and Techniques, Simon Fraser University, 2000

2 G. V. Trunk, A problem of Dimensionality: A Simple Example, IEEE Trans. Pattern Anal. Mach. Intelligence, vol. 1, pp. 306-307, 1979

3 A. K. Jain and R. Dubes, Feature definition in pattern recognition with small sample size, Pattern Recognition, vol. 10, pp. 85-97, 1978

4 F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In: Pattern Recognition in Practice IV, Multiple Paradigms, Comparative Studies and Hybrid Systems, eds. E. S. Gelsema and L. S. Kanla. Amsterdam: Elsevier, 1994. pp. 403-413.

5 S.Cost and S. Salzberg, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," Machine Learning, Vol.10, No. 1, Jan. 1993, pp. 57-78.

6 W. F. Punch, E.D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody, "Further research on feature selection and classification using genetic algorithms," in Proc. Int. Conf. Genetic Algorithms, 1993, pp. 557-564

7 J. D. Kelly and L. Davis, "Hybridizing the genetic algorithm and the K nearest neighbors classification algorithm," in Proc. 4th. Int. Conf. Genetic Algorithms Appl., 1991, pp.377-383

8 G. John, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," Proc. 11th Int'l Conf. Machine Learning, Morgan Kaufmann, San Fransisco, 1994, pp. 121-129

9 H, Liu and R. Setiono. "A Probabilistic Approach to Feature Selection- A Filter Solution," Proc. 13th Int'l Conf. Machine Learning, Morgan Kaufmann, 1996, pp. 319-327

10 F. Brill, D. Brown and W. Martin, "Fast Genetic Selection of Features for Neural Network Classifiers," IEEE Trans. Neural Networks, Vol. 3, No. 2, Mar. 1992, pp. 324-328.

11 M. Richeldi and P. Lanzi, "Performing Effective Feature Selection by Investigating the Deep Structure of the Data," Proc. Second Int'l Conf. Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, Calif, 1996, pp. 379-383.

12 A. S. Fraser, "Simulation of genetic systems by automatic digital computers-I: Introduction," Australian J. Biol. Sci., vol. 10, pp. 484-491, 1957

13 J. L. Crosby, "Computers in the study of evolution, " Sci. Prog. Oxf., vol. 55, pp. 279-292, 1967

14 H. J. Bremermann, M. Rogson, and S. Salaff, "Global properties of evolution process," in Natural Automata and Useful Simulations, H. H. Patte, E. A. Edlsack, L. Fein, and A. B. Callahan, Eds. Washington, DC: Spartan, 1966, pp. 3-41.

15 J. Reed, R. Toombs, and N. A. Barricelli, "Simulation of biological evolution and machine learning: I. Selection of self-reproduction numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing," J. Theoret. Biol. , vol. 17, pp. 319-342, 1967, IEEE International, 3: 2140-2142, 2000.

16 D.B. Fogel, Ed., Evolutionary Computation: The Fossil Record. New York: IEEE Press, 1998

17 J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. Michigan Press, 1975.

18 D. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989

19 R. E. Marmelstein, "Evolving Compact Decision Rule Sets", PhD. Dissertation, Air Force Institute of Technology, 1999.

20 R.O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, New York: Wiley, 1973

21 Kira, K., and L. A. Rendell, "A Practical Approach to Feature Selection", in Proceedings of the Ninth International Conference on Machine Learning, pp. 249-256, 1992

22 Krivda, C. D., "Data-Mining Dynamite," BYTE, pp. 97-102, October 1995

23 Cuadrado, J. L., "Mining Statistics," BYTE, pp. 97-100, February 1995

24 P. Larranaga, C.M.H. Kujpers, R.H. Murga I. Inza and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representation and Operators"

25 Wagner, T.J. "Convergence of the edited nearest neighbor," IEEE Transactions on the Information Theory, IT (19) (Sept 1973)

26 Devijer, P.A. and J. Kittler. "On the Edited Nearest Neighbor Rule." Proceedings of the 5th International Conference on Pattern Recognition, pp. 72-80. December 1980.

27 R. Jain, "The Art Of Computer Systems Performance Analysis", John Wiley & Sons Inc., pp.207, 1991

28 A. S.Wu, R.K. Lindsay, R. L. Riolo, "Empirical observations on the roles of crossover and mutation", Proceedings of Seventh International Conference on Genetic Algorithms, pp. 362-369. July 1997.

29 T. Back, D.B. Fogel, Z. Michalewicz, "Evolutionary Computation 2 Advanced Algorithms and Operations", Institute o Physics Publishing, pp.56-57, 2000

30 Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolutionary Programs", pp. 179-180, 1994

31 Srinivas, N. and Deb, K., Multi-objective Optimization Using Non-dominated Sorting in Genetic Algorithms, Department of Mechanical Engineering, Indian Institute of Technology, Kanput, India, 1993

32 Schaffer, J.D., Some experiments in Machine Learning Using Vector Evaluated Genetic Algorithms, PhD Dissertation, Vanderbilt University, Nashville, 1984

33 Grefensette, J.J. GENESIS: A System for Using Genetic Search Procedures, Proceedings of the 1984 Conference on Intelligent Systems and Machines, pp. 161-165

34 David Strong, "Implementation and Analysis of the Parallel Genetic Rule and Classifier Construction Environment", Master's Thesis, Air Force Institute of Technology, 2001

35 J. Sklansky and W. Siedlecki. A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters, 10:335-347, December 1989.

36 Christopher M. Bishop. Neural Networks for Pattern Recognition. Oxford Univesity Presss, 1995

37 Chulhee Lee and David Landgrebe. Feature extraction based on decision boundaries. IEEE Transactions on Patterns Analysis and Machine Intelligence, 15 (4), April 1993.

38 J. Aguilar, J. Riquelme y Miquel Toro, “Three Geometric Approaches for representing Decision Rules in a Supervised Learning System”, Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference, pp. 8, July 1999

# VITA

1st. Lt. Okan YILMAZ was born on 30 September 1976 in Kayseri, Turkey. He graduated from Maltepe Military High School in Izmir, in August 1994. He entered undergraduate studies at the Air For Academy in Istanbul where he graduated with Bachelor of Engineering degree in Computer Engineering in August 1998. After graduating from Air Force Academy, he entered the Air Technical School in Izmir, in September 1998 and graduated in August 1999 as a communication officer. His first assignment was at Dalaman AFB as a communication officer in October 1999. In August 2000, he entered the Department of Electrical and Computer Engineering, School of Engineering, Air Force Institute of Technology.

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 074-0188* |
|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* 18-03-2002 | **2. REPORT TYPE** **Master's Thesis** | **3. DATES COVERED** *(From – To)* Jun 2001 – Mar 2002 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** DATA MINING FEATURE SUBSET WEIGHTING AND SELECTION USING GENETIC ALGORITHMS | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Yilmaz, Okan, Lieutenant, TUAF | **5d. PROJECT NUMBER** |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)** Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 P Street, Building 640, WPAFB OH 45433-7765 | **8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCE/ENG/02M-05 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Dr. Robert L. Ewing, Tech Adv., AFRL/IFTA, WPAFB, OH 45433, Ph. (937) 255 66 53 #3592, Robert.Ewing@wpafb.af.mil | **10. SPONSOR/MONITOR'S ACRONYM (S)** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

Professor Gary B. Lamont, CIV, AFIT/ENG, Ph. (937) 255-3450

**14. ABSTRACT**

We present a simple genetic algorithm (sGA), which is developed under Genetic Rule and Classifier Construction Environment (GRaCCE) to solve feature subset selection and weighting problem to have better classification accuracy on k-nearest neighborhood (KNN) algorithm. Our hypotheses are that weighting the features will affect the performance of the KNN algorithm and will cause better classification accuracy rate than that of binary classification. The weighted-sGA algorithm uses real-value chromosomes to find the weights for features and binary-sGA uses integer-value chromosomes to select the subset of features from original feature set. A Repair algorithm is developed for weighted-sGA algorithm to guarantee the feasibility of chromosomes.

By feasibility we mean that the sum of values of each gene in a chromosome must be equal to 1. To calculate the fitness values for each chromosome in the population, we use K Nearest Neighbor Algorithm (KNN) as our fitness function. The Euclidean distance from one individual to other individuals is calculated on the d-dimensional feature space to classify an unknown instance. GRaCCE searches for good feature subsets and their associated weights. These feature weights are then multiplied with normalized feature values and these new values are used to calculate the distance between features.

**15. SUBJECT TERMS**

Genetic Algorithm, Data Mining, Classification, Feature Subset Weighting and Selection, K Nearest Neighbor Algorithm

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Lamont, B. Gary, Professor, CIV, AFIT/ENG |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | **19b. TELEPHONE NUMBER** *(Include area code)* |
| U | U | U | UU | 124 | (937) 255-3450, ext 4718; e-mail: Lamont.Gary@afit.edu |