MFS-PSO: A modified PSO method for optimizing gene selection with dynamic adaptation employing a boosting approach

Authors

Arif Muhammad Sultan — Student Id: 084404

Nabil Bin Hannan — Student Id: 084412

Supervisor

Prof. Dr. M. A. Mottalib

Head of the Department

Department of Computer Science & Engineering (CSE)

Islamic University of Technology (IUT)

Co-Supervisor

Shaikh Jeeshan Kabeer

Lecturer

Department of Computer Science & Engineering (CSE)

Islamic University of Technology (IUT)

A Thesis submitted to the Department of Computer Science & Engineering (CSE)
in Partial Fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science & Engineering (CSE)

Department of Computer Science & Engineering (CSE)

Islamic University of Technology (IUT)

Organization of the Islamic Cooperation (OIC)

Gazipur, Bangladesh

September, 2012

CERTIFICATE OF RESEARCH

This is to certify that the work presented in this thesis paper is the outcome of the research carried out by the candidates under the supervision of Prof. Dr. M. A. Mottalib, Head of the Department, Department of Computer Science and Information Technology, IUT and co-supervision of Shaikh Jeeshan Kabeer **,**Lecturer, Department of Computer Science and Information Technology, IUT, Gazipur. It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree or any judgment.

Authors

Arif Muhammad Sultan

Nabil Bin Hannan

Signature of Co-Supervisor

Shaikh Jeeshan Kabeer
Lecturer
Department of CSE, IUT

Signature of Supervisor

Prof. Dr. M. A. Mottalib
Head, Department of CSE, IUT

Signature of the Head of the Department

Prof. Dr. M. A. Mottalib
Head, Department of CSE, IUT

Abstract

Microarray technology development have meant that the dimensionality of data that is produced by the Microarray chips have increased many folds over the years. Pattern recognition and other subsequent analysis from the thousands of gene expression values is particularly difficult and primary role of an effective feature selection is to simplify this task. Removal of less informative genes helps to alleviate the effects of noise and redundancy, and simplifies the task of disease classification and prediction of medical conditions such as cancer. In this study the shortcoming of the current PSO based approach for feature selection has been improved. A boosted filter and wrapper models are put to use to take advantage of the facilities that each provides. As filter method exhibits some limitations, in this study a boosted approach to filtering (BFSS) has been employed. BFSS iteratively selects genes in each iteration and emphasizes on the misclassified samples and in subsequent iterations it tries to find effective genes for the misclassified samples. This allows BFSS to perform better than traditional Filter methods as it focuses on its weaknesses. Traditional PSO based methods and other similar approaches suffer primarily from over fitting problem and the initial population is large and random. The gene subset provided by BFSS is fed to a Particle Swarm Optimizer (PSO) which reduces the feature subset in smaller numbers at each iteration. This helps to generate a better optimal subset of genes. The proposed hybrid approach is applied on leukemia, colon and lung cancer benchmarked datasets and have shown better results than other well-known approaches.

Table of Contents

Chapter 1

Introduction:

1.1 Overview

Modern Technology has blessed us by many folds but with increasing dimensionality of various data. Thus, to take a grip of this massive amount of data, several methodologies have been applied and still many researches are going on. The main focus in recent days is to minimize the space and time complexity to get a better solution in several critical research areas. Machine learning can greatly help as computer can swiftly process huge amount of data within a short period of time. But the main concern is to create some efficient model to process this data.

Our thesis concerns on feature selection. Feature selection is the technique of selecting a subset of relevant features for building robust learning models [1]. Our initial study was based on all the implemented evolutionary algorithms. We have also done several comprehensive analyses between the existing methods. Moreover, we had an extensive comparison about the advantages and drawbacks of the existing algorithms. There are many areas where feature selection is very important like robotics, pattern recognition, text categorization, medical diagnosis, combinatorial chemistry and many more.

The relentless development of Microarray datasets [2, 12] have meant that the dimensionality of data that can be represented by the Microarray chips have increased many folds over the years. Usually features that are analyzed may be thousands in number, whereas the number of samples is very less hovering around lower hundreds.

Feature selection attempts to identify and highlight the most informative genes in the microarray data sets which have dominant effects on the biological states of human cells. Our main focus is to select the most informative genes to optimize the dimensionality problem. Removal of less informative genes helps to alleviate the effects of noise and redundancy. Moreover, it simplifies the task of disease classification and prediction of medical conditions such as cancer.

Feature selection techniques can be classified into three broad categories: Filter Technique, Wrapper Model and Embedded Technique. Filter technique performs individual gene selection process independent of the classification model. Ranking, signal-to-noise or information gain are the more popular filter models which consider genes by one-by-one basis.

1.2 Problem Statement

The main problem of high dimensional data is the inclusion of noisy and irrelevant data in the information set. As the datasets become large the number of noisy, redundant and uninformative gene also increases resulting in space-time complexity. However, if we apply feature selection a decrease in space complexity, a cost reduction of feature measurement and an increase of classifier accuracy and efficiency can be achieved

1.3 Research Challenges

The high dimensionality results in an immense feature space and thus execution of a brute force exhaustive search should not be encouraged. Therefore, to achieve an accurate and efficient evaluation of samples an optimal method needs to be devised. The desired outcome of the method is minimizing the number of features and increasing the predictive power of the classifiers. To add more intensity to the problem domain this field of bioinformatics produces inadequate testing and training samples. Along with the removal of noisy, irrelevant and redundant information the proposed method must be able to handle the correlation factor existing between the features and thus utilize the combined predictive power. This study encompasses all these factors and theoretically expects to bring about better results.

1.4 Motivation

This study aims at deriving a better method for feature selection using an evolutionary approach. This approach has an upper hand on other approaches as it does not require considering the entire feature space. An optimal configuration of the evolutionary approach may help get over the generalization and over fitting problems.

1. 5 Scopes

Our study has an extensive scope area where evolutionary algorithm is a recent approach for feature selection. In case of research, to improve the search criteria for feature selection two probable approaches can be taken. Firstly, the improvements can come from existing approaches. Secondly, it may come from generating new approaches. In our case, we are going to propose a hybrid approach which is the combination of two algorithms. This approach tends to reduce some shortcomings of the existing methods which we will discuss later. Feature selection will help other working fields' ex. Pattern recognition, Computer vision, Artificial intelligence etc.

1.6 Research Contribution

In this study the limitations of the current PSO based approach has been improved. A boosted filter and wrapper models are put to use to take advantage of the facilities that each provides. The original dataset would be reduced to about half by using the t-score filter method. As filter method on its own has shown to introduce redundant features, in this study a boosted approach to t-score (BFSS) [5] has been employed. However BFSS cannot handle the noise introduced in the dataset, in addition it does not consider the classification accuracy of its selection and cannot fully utilize the predictive power of a combination of genes. To overcome the shortfalls of BFSS we will use Particle Swarm Optimization.

1.7 Thesis Outline

In Chapter 1 we have talked about the introduction of our study in a précised manner. Chapter 2 deals with the basic feature selection method and some highlighted evolutionary approaches with a brief discussion about PSO method. Chapter 3 will be discussed about our proposed algorithm and some elaborate discussion. Chapter 4 will consist of the experimental analysis and result comparisons.

Chapter 2

Literature Review:

2.1 Feature Selection

In case of our study, feature selection is the technique of selecting a subset of relevant features (genes) for building robust learning models [5]. With the passage of time processing speed of computer has increased and also a lot of data collection technologies have been improved. But the data that is generated is really enormous which cannot be handled efficiently in a short time. However, if we use feature selection method we can optimize the huge dataset into a smaller version that deals somewhat with the space-time complexity problem.

Nowadays feature selection techniques are widely used in statistics and machine learning. We focus on the second part, where the performance of the algorithm can be automatically improved through experience. Machine learning algorithms are organized into a taxonomy based on the desired outcome of the algorithm.

- **Supervised learning** generates a function that maps inputs to desired outputs.
- **Unsupervised learning** models a set of inputs.
- **Semi-supervised learning** combines both labeled and unlabeled examples to generate an appropriate function or classifier.
- **Reinforcement learning** learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.
- **Transduction** tries to predict new outputs based on training inputs, training outputs, and test inputs.
- **Learning to learn** learns its own inductive bias based on previous experience.

Feature selection method focuses on supervised learning and we are also using this approach in our study to construct an effective learning model.

2.2 Feature Selection Techniques

Feature selection can be applied on a set of features which can be a better solution than choosing all possible subsets of features [14]. It is impractical if a large number of subsets are available. Feature selection can be classified into two broad categories:

1. *Feature Ranking (FR):*

This is also known as feature weighing which assesses individual features and assigns them weights according to their degree of relevance. Many researches have been done with feature ranking as the base method (for example Bekkerman et al., 2003, Caruana and de SA, 2003, Weston et al., 2003).

2. *Feature Subset Selection (FSS):*

This technique measures the goodness of each found feature subset. A great deal of work has also been done Feature subset selection (for example Guyon et al., 2004, Ma & Huang, 2005, Ooi & Tan 2003).

FSS is more effective than FR technique because Feature Ranking (FR) uses the predictive powers of individual features whereas FSS utilizes the power of a specific set of features. We are going to apply FSS technology which will result in a more optimized solution. FSS follows 3 basic methodologies:

- **Filters:**
  Filter techniques takes in account the relevance of features by looking only into the intrinsic properties of data. A feature relevance score is calculated for example in signal-to-noise is used for scoring genes. From the scored set of genes the low scoring features are removed and the remaining subset features with the higher scores are presented to the classifier algorithm for classification. Similarly in information gain which is another scoring method is used to score the genes from which subsequent removal of low scoring genes is done. Besides these F score, t score and correlation of genes are used for scoring.

- **Wrappers:**
  In this technique a search in the entire feature space is performed to generate a subset of features which are evaluated against a classification model in which the classifier is integrated to evaluate the effectiveness of the subset of features generated. In Genetic Algorithm is used for selecting and SVM classifier was

used for evaluating the accuracy of the results [4]. Similarly in evolutionary approach is used for selecting the features and MLHD (Most Likelihood Classification Method) classifier is used for evaluating the choices made.

- **Embedded Methods:**
  This technique searches in the entire feature space to search in order to find optimal subsets of features. The search procedure is built into the classifier in this variation. **[**Duda et al. 2001] used Bayesian theory to guide the search process based on the probability of selection. In SVM classifier was over hauled to weigh features and adjust them for making selection in the process of finding the solution.

2.2.1 Feature Searching

In order to perform feature selection the feature space needs to be traversed i.e feature searching. Feature searching involves going through the feature space to select features to be used for classification. Many approaches to feature selection exist which can be broadly classified into the following:

Exhaustive

Brute-force search or exhaustive search is a simple but very general problem-solving technique that methodically traverses all the possibilities for the solution, checking whether each of the candidates satisfies the solution criteria. Exhaustive search is easy to implement and will guarantee a solution if it exists. The downside is that its cost is proportional to the number of candidate solutions, which tends to grow very rapidly as the size of the problem increases. Thus this approach is usually used when the problem size is limited

Best first

It is a heuristic searching technique primarily used on graphs. As it traverses through the candidate solution it selects the one which appears to be the best choice under the current situation and moves forward. In this manner searching continues. However this approach does not ensure an optimum solution. The selection of a particular candidate is defined by an evaluation function. The particular evaluation function used to determine the score of a node is not precisely defined in the above algorithm, because the actual function used is up to the determination of the programmer, and may vary depending on the particularities of the search space. While the evaluation function can determine to a large extent the effectiveness and efficiency of the search.

Simulated annealing

In metallurgy annealing is the process used to temper or harden metals or glass by heating them to a high temperature and then gradually cooling them thus allowing the material to combine into a low energy crystalline structure. In simulated annealing instead of choosing the best move it picks a random one and if the move improves the situation then it is accepted otherwise the probability of the move is decreased to ensure that such moves are not chosen in the future. As the probability of the badness of the moves decreases so does the chances of it being chosen again. Also the probability decreases with time.

Greedy approach

A greedy approach makes locally optimal choice at each stage of the searching as the search moves towards finding a global optimum. For example in a traveling salesman problem "At each stage visit the unvisited city nearest to the current city" approach is followed. In general, greedy algorithms are used for optimization problems. Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. They can make commitments to certain choices too early which prevent them from finding the best overall solution later. Greedy algorithms can be characterized as being 'short sighted', and as 'non-recoverable'. They are ideal only for problems which have 'optimal substructure'. Despite this, greedy algorithms are best suited for simple problems.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary method which simulates the social behavior of living organisms like: bird flocking and fish schooling. Here candidate solution moves around a D-dimensional search space. Each solution is termed as a chromosome and collection of such chromosomes is called a population. PSO continuously updates it’s generation at every iteration. From the population a fitness function defines the goodness or “fitness” of solutions. Each particle uses its individual memory and swarm knowledge to find the best solution. At every iteration particle value is renewed based on its fitness value.

Exhaustive approach involves evaluating each and every feature before a selection is made which is computationally very expensive. Best first searching halts at the encounter of the best match encountered. Simulated annealing is the minimization of the learning process based on successive update steps where the update step length is proportional to an arbitrarily set parameter. Particle Swarm Optimization has memory of its own, so knowledge of good solutions is retained by all the particles and an optimal solution can be found by the swarms following the best particle. Unlike other approaches, in PSO only the global best value gives out the information to others. It means every particle has the ability to share its information with the other particles. Computation time used in PSO is less than other evolutionary algorithms.

In this study the feature selection technique is being studied and not the classifiers. The standard classifiers which have been well tested and being widely used will be used for classification purposes. In order to choose the features which will be best for classification the help of evolutionary algorithms will be taken. In order to gain a deep understanding of the evolutionary computing a variant of the same, Particle Swarm Optimization was exhaustively studied along with its variants which is covered in the following section.

2.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are motivated by the theory of evolution as proposed by Charles Darwin. Evolutionary algorithms are based on a simplified model of this biological evolution. An environment in which the potential solutions can evolve is created is created. The environment is defined by parameters which are related to the problem being solved. Evolutionary Computation comprises of several variants of algorithm which includes Particle Swarm Optimization (PSO), Genetic Algorithms (GAs), Evolution Strategies, Genetic Programming (GP), Evolutionary Programming and Learning Classifier Systems. The most advantageous type of evolutionary algorithm is the Particle Swarm Optimization technique.

- **PSO:** Particle Swarm Optimization is a method that optimizes a problem iteratively. PSO treats each solution as a particle and starts with a pool of candidate solutions. The particles are moved in the search space and the movement of each particle is guided by the best known local position. The process is iterated as each local solution is found to guide the solution towards the global best position or intended optimal solution.

- **GA:** Genetic Algorithm is a type of Evolutionary Algorithm inspired by the biological method of evolution in which an environment is created in which potential solutions can evolve. From the population a fitness function selects some solutions based on its “goodness” which are subjected to genetic operators such as: Mutation and Crossover which generates new population. From this population the entire process is repeated until the optimal solution has been found.

- **ACO:** Ant Colony Optimization relies on a probabilistic model so solve problems. The original algorithm was used to find the best path in a graph[9]. The algorithm was later modified to solve a wide class of problems across various applications.

- **DE:** Differential Evolution method optimizes a problem by iteratively improving a candidate solution with respect to some given measure of quality[6]. DE maintains a collection or population of candidate solutions and it creates new solutions by combining existing ones and keeping the solution which has higher fitness or goodness values.

In our study we have focused on the Particle Swarm Optimization algorithm. The existing PSO method works more or less moderate but with some shortcomings. The next sections describe basic PSO and the exhaustive study.

2.3.1 Particle Swarm Optimization (PSO):

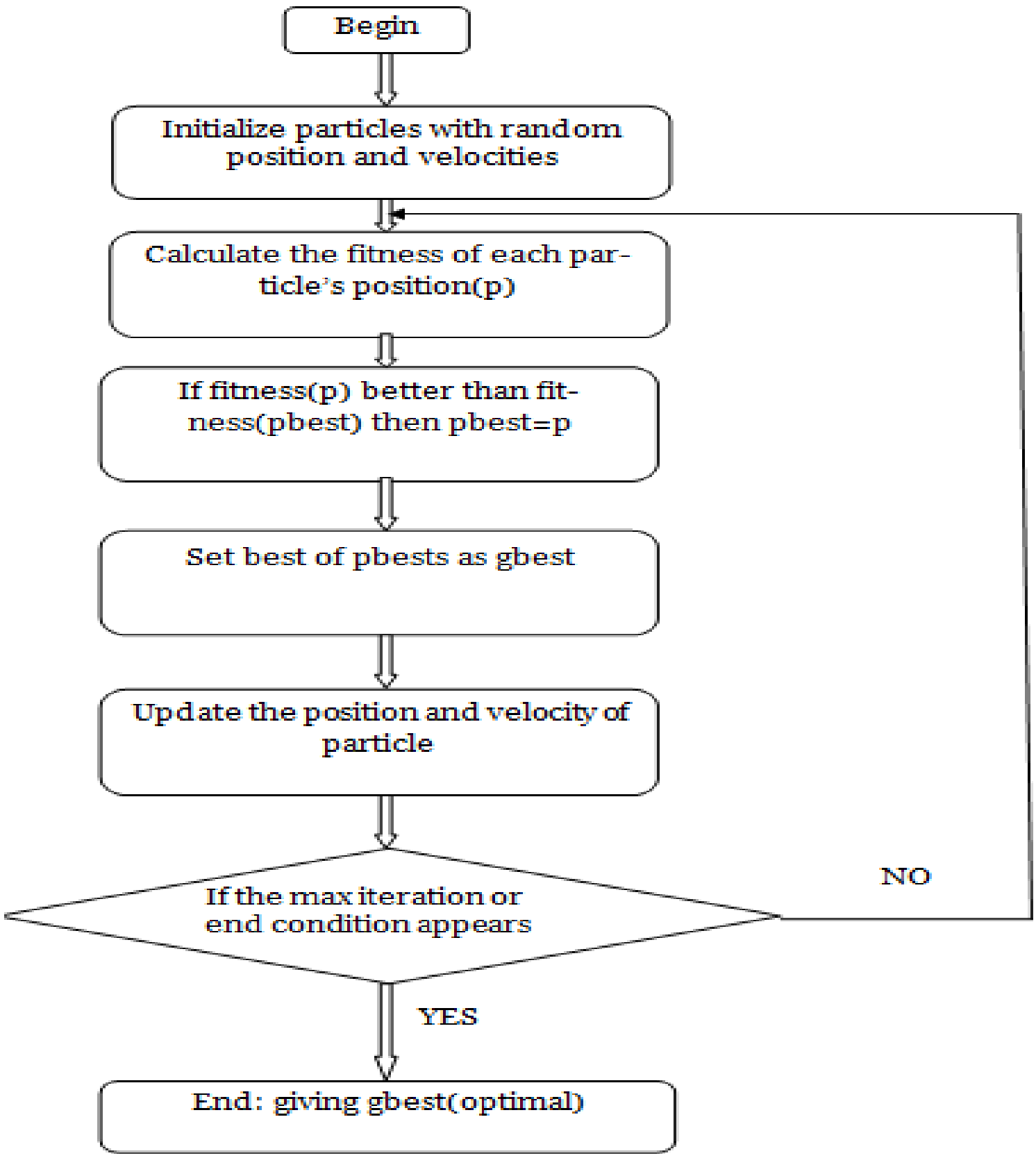
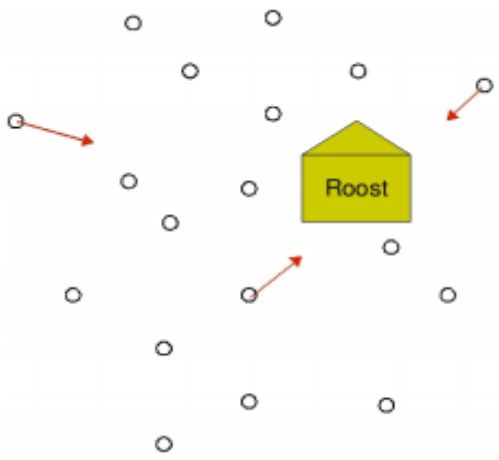


Fig 1: Basic flowchart of PSO

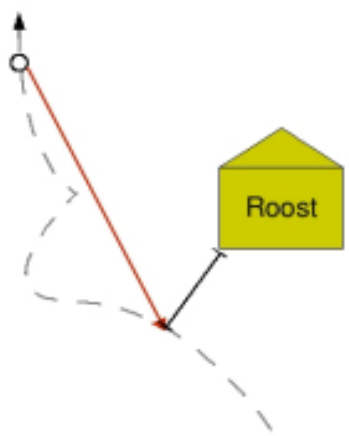
PSO is one of the evolutionary algorithms [3] which imitate the behavior of organisms such as: bird flocking or fish schooling. In PSO, each single candidate solution is "an individual bird of the flock", that is, a **particle** - search space. In our field it represents a feature subset. Here the term 'swarm' denotes the group of particles which collectively share information about the best solution.

PSO was first found by Kennedy and Eberhart who included a 'roost' in a simplified Reynolds-like simulation so that:

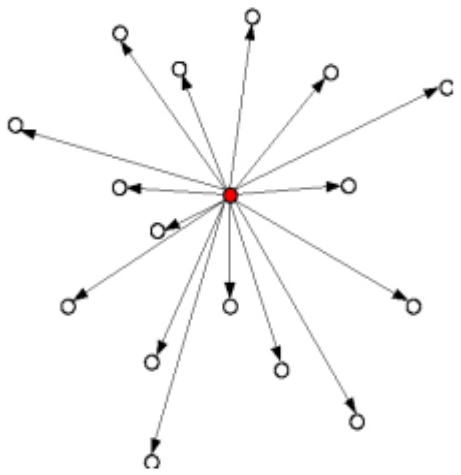
I. Each agent was attracted towards the location of the roost.



II. Each agent 'remembered' where it was closer to the roost.



III. Each agent shared information with its neighbors (originally, all other agents) about its closest location to the roost.



2.3.2 Swarms and Particles:

Millonas articulated five basic principles of swarm intelligence in his model for application in artificial life [17]:

1. ***Proximity Principle:***
   The population should be able to carry out simple space and time computations.

2. ***Quality Principle:***
   The population should be able to respond to quality factors in the environment.

3.Diverse Response:

Population should not commit its activities along excessively narrow channels.

4.Principle of Stability:

Population should not change its mode of behavior every time the environment changes.

5.Principle of adaptability:

The population must be able to change its mode of behavior when its worth the computational price.

The calculations over the D-dimensional search space are carried out over several time steps or iterations. The adaptive functions, that is the fitness values pbest and gbest which are the quality factors, stimulates the population to change its behavior. The allocation of population responses between pbest and gbest ensures a diversity of response. Conforming to the principle of stability the population changes its mode of behavior *only* when gbest value changes. The population is adaptive because it *changes* when gbest changes.

2.3.3 Finding the best solution:

Each particle makes use of its individual memory and knowledge gained by the swarm as a whole to find the best solution PSO simulates the social behaviour of organisms, such as bird flocking and fish schooling. In PSO, each single candidate solution is "an individual bird of the flock", that is, a particle - search space In our field it represents a subset of features.

All of the particles have fitness values, which are evaluated by fitness function to be optimized, and have velocities which direct the movement of the particles

During movement each particle adjusts its position according to its own experience, as well as according to the experience of a neighboring particle and makes use of the best position encountered by itself and its neighbor.

The particles move through the problem space by following a current of optimum particles.

The initial swarm is generally created in such a way that the population of the particles is distributed randomly over the search space.

At every iteration each particle is updated by the following two "best" values, called pbest and gbest.

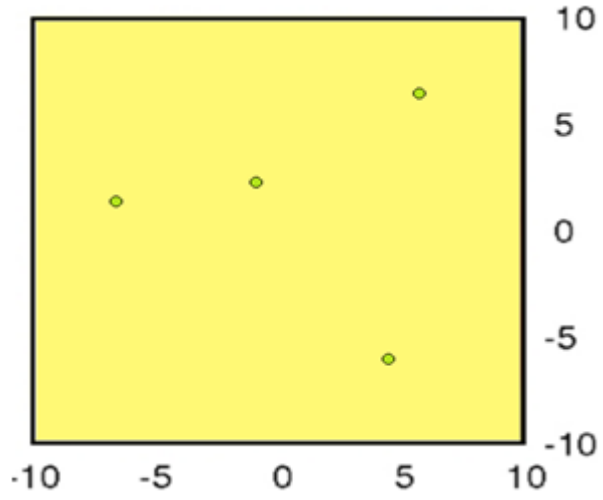
P - Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) the particle has achieved so far. This fitness value is stored, and called pbest. The best pre visited position of the i-th particle is denoted its individual best position pi = (pi1, p 2 , ..., piD), a value called p*best*i.

G -When a particle takes the whole population as its topological neighbor, the best value is a global "best" value and is called gbest. The best value of all the individual p*best*i values is denoted as the global best position g = (g1, g2 , …, gD) and called g*best*.

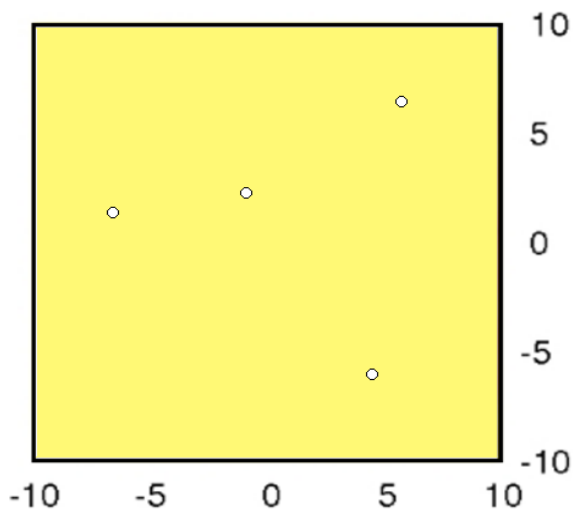
2.3.4 Visualisation of the PSO Procedure

1.

The particles in the swarm represents candidate solutions. Each particle is an logical representation of a feature subset that is, a chromosome. From among thousands of features (genes) several subsets of genes are created randomly and uniformly distributed over the D-dimensional search space. Each subset is known as particle and all the particles collectively form the swarm.

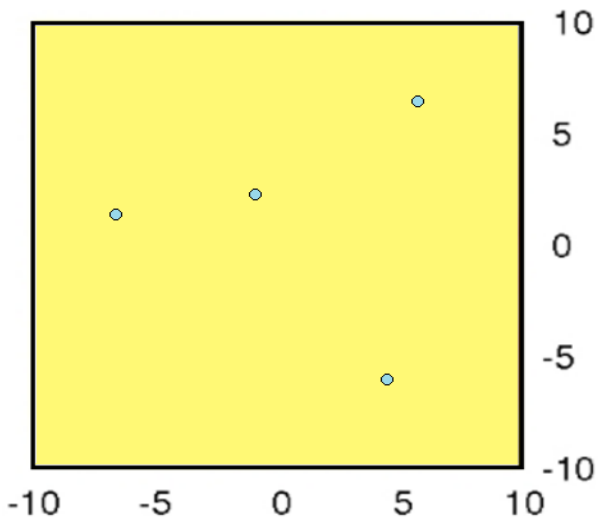


2.



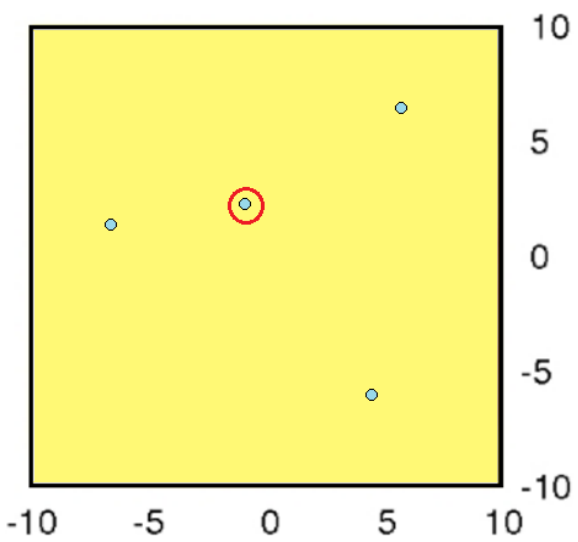
3.

Each of particle has a fitness value which is calculated each time a new population is created and a new generation is initialized. Whenever a particle comes across a local best fitness value which is higher than its previous best solution, it immediately updates the variable **pbest** which is the best fitness value the particle has achieved so far.



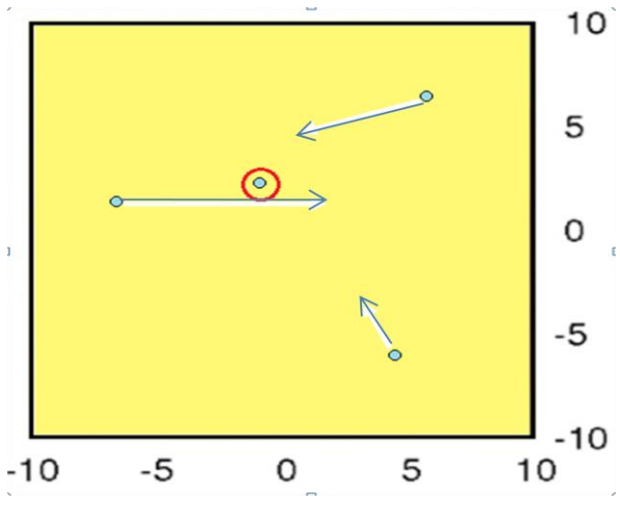
4.

In each iteration, for every particle the gbest value is determined which is the best fitness value of all the pbest values of the particles in its neighborhood that is generally considered as the whole swarm.



5. = w + (- + × (—) .

The initial velocities of each of the particles was given zero but as soon as the gbest and the pbest values of a associated with a certain particle is identified it gains a velocity towards the optimum solution according to the above equation where is the final velocity , is the current velocity , is the local best solution, is the current position and is the global best solution. , are acceleration constants and w is the inertia weight.



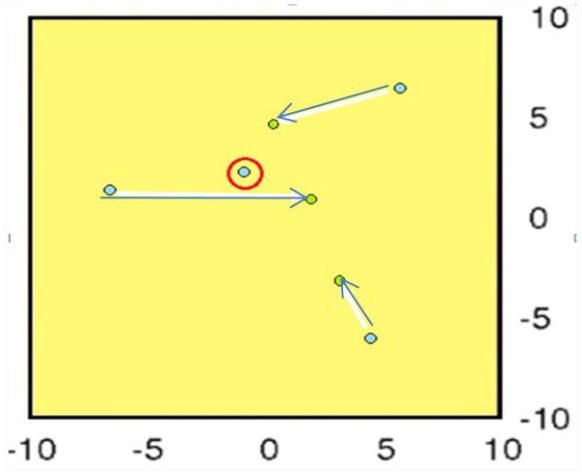
6. Move particle to their new position according to = +

The positional renewal of the particles is implemented with the following equations :

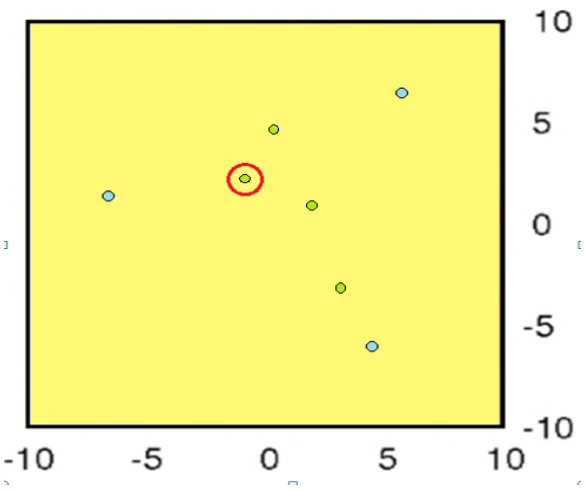
S( ) = ———

if (< S( )) then = 1; else = 0

where S( ) is a feature calculation function and if its value is more than a random variable then the corresponding feature is set as 1, meaning that it is selected for next update and if it is less than then it is set as 0 meaning that the feature is not selected.



7. :



These steps are repeated until maximum number of generations are reached or an optimal solution is found.

2.3.5 Some Existing Shortcomings of PSO

In spite of being an easy to compute and an easy to implement algorithm the basic PSO approach exhibits some drawbacks [13, 16] namely:

1. Local optima problem :

Sometimes it is easy to be trapped in local optima /The velocity of particles rapidly approach zero as it gets closer to an nearly optimal solution and the convergence rate decreases considerably in the later period of evolution.

2.

The search process of the PSO is non-linear and very complicated.

3. Random Initialization:

In PSO the particles are initiated randomly using all the genes present in the dataset. Thus, irrelevant genes would increase the space and time complexity

In our proposed approach we will be dealing with two of the shortcomings which will be discussed in the next chapter.

Chapter 3

Proposed Method

3.1 Overall Concept

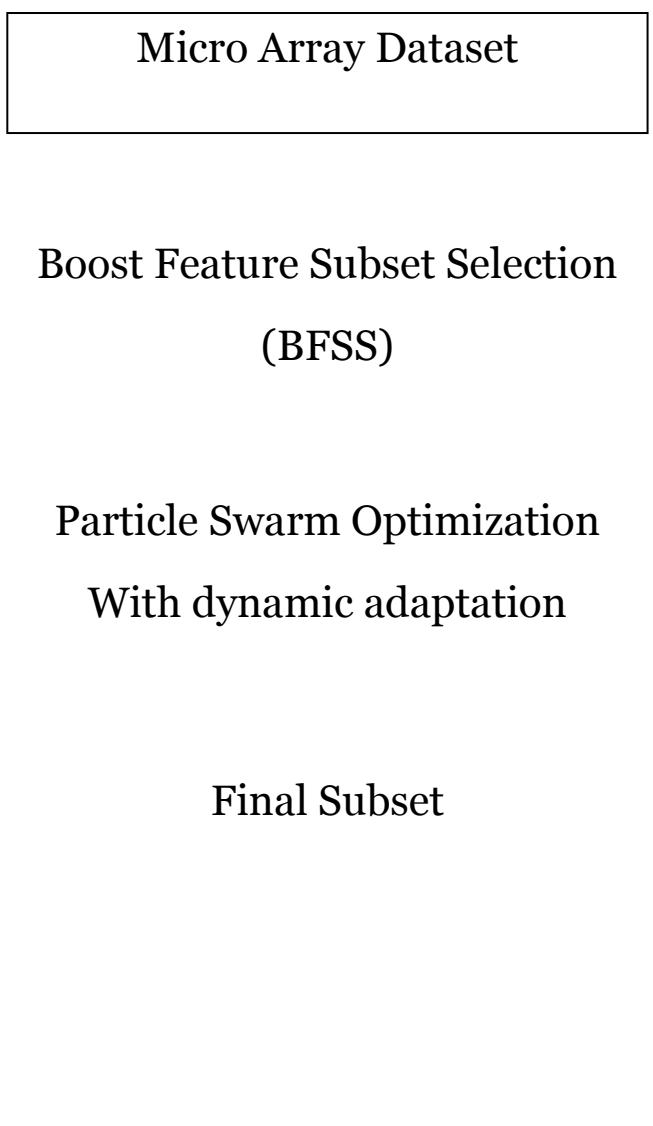


Fig 2: Proposed Approach

Minimized Feature Space(MFS) PSO :

The basic steps of our proposal is outlined as follows :

1. Micro Array outcome is in the form of a matrix which represents gene expression data.

2. Then we are using BFSS to minimize the dimensionality of feature space considering the sample scores.

3. We propose to apply MFS-PSO to get the final best gene subset. This approach **overcomes local optima problem** using dynamic adaptation approach by dynamically changing inertia weight (w) value.

3.2 Boost Feature Subset Selection

Boosting means producing a very accurate prediction rule by combining rough and moderately inaccurate "rules of thumb". In the context of single gene based feature selection boosting would improve the performance by identifying the weak performers in a particular iteration and in the successive iterations it would try to identify features that would perform well for those weak performers. The performance of the process is said to be “boosted” by giving more emphasis on the weak performers in a particular iteration as shown in Xian Xu and Aidong Zhang.

- **Illustrating example of redundancy in selected gene set**

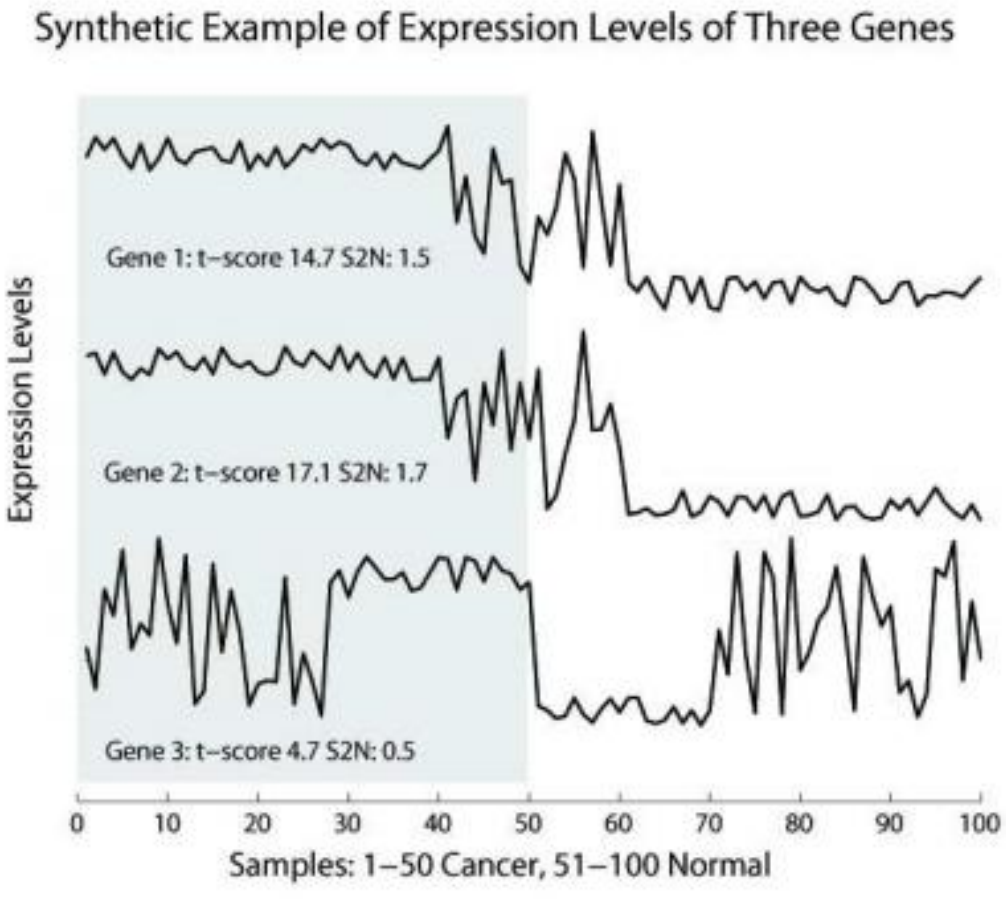


Fig 3: Example of redundancy in selected gene set

The first two genes, gene 1 and gene 2 behave similarly. In majority of the samples (samples 1 to 40 and samples 61 to 100, or 80% of samples), the expression levels of gene 1 or gene 2 can be used to predict sample class labels effectively. Actually the expression levels of these two genes are generally higher in cancer samples than in normal ones. However, the expression levels of these two genes in samples 41 to 60 (20% of samples) are more mixed across cancer/normal class distinction.

- **<u>BFSS –some terminologies</u>**

First we will discuss about some basic terminologies of BFSS before moving to the detailed description-

- A bootstrap sample set is a multiset of samples randomly drawn with replacement from the original set of samples S. The sampling probability of each sample in S is determined by a probability table p(s) where s∈ S.

- The worst set of samples with respect to bootstrap dataset B and a single-gene based scoring function F is defined as a multiset:

$$(F(E(g, -S)))$$

And $|S|=\partial$

Here - S means a set by removing S from . We also call - the best set of samples.

Boost Feature Subset Selection (BFSS) algorithm starts off by generating a set of samples called a bootstrap sample set which is a multiset of samples obtained by random sampling from the pool of all samples S. The probability of a sample being selected is equal to p(s) where s ∈ S and initially all samples have a probability of – .

- **<u>BFSS (Steps</u>):**

1. Score all the genes in the dataset, sort them and select the top score gene.
2. Worst set of sample is identified.
3. Probabilty of these samples for selection in next iteration is increased.
4. Next iteration selects gene performing well for the worst sample set

5. BFSS would run until the number of selected genes (BG) has been found which depends on the dataset being evaluated.

Algorithm **1: Boosted Feature Subset Selection**

n is the number of genes to be selected. F is a single gene based discriminative score

1. Initialize ***P(s)*** to be — (is the total number of samples in the dataset).Set

2. as an empty set

3. For | | < n do

4. Generate the bootstrap sample set

5. Calculate the F score on bootstrap, keep track of the best score so far

6. Add top ranked gene *g* based on F score to

7. Find worst ∂ samples based on gene *g* and using algorithm 2

8. Reduce the probability for the *best set of samples* (those samples which are classified accurately by the gene *g*)

9. Remove *g* from dataset

End for

10. Return

Algorithm 2: ***Worst Sample Set***: calculate the worst set of samples

1. , to be empty sets
2. For all s in *S* do
3. $S - \{s\}$
4. Calculate F score for the gene *g,* add score to
5. End for
6. Sort, , add samples *S* corresponding to top ∂ scores in , to
7. Return

- **Pseudocode explanation**

The Boost feature subset selection algorithm (BFSS) is depicted by Algorithm 1 and Algorithm 2. After initialization, the Algorithm 1 generates a bootstrap of training set using random sampling with replacement from S using probability table p(s). After bootstrap produced, the F score is calculated for each gene in . The gene with best F score for the current is selected and added to the selected gene

set . Based on this best gene g, BFSS ascertains the worst set of samples with respect to g and the single-gene based scoring function F using Algorithm 2. The probability table p(s) which affects the generation of further bootstrap is updated by reducing the probabilities of the non-worst or good samples. Thus the probability of selecting these good samples being in the later iterations is thus reduced, causing BFSS to shift the focus onto those samples that previously selected genes would not perform well on i.e the worst set of samples for the current *g*. The currently selected gene is then marked as selected and hence it is not considered further by the BFSS algorithm. BFSS repeats this process until *n* genes are selected. Experimentally *d* was chosen to be 0.96 of the number of training samples in a dataset and e to be 0.96.

After implementing the BFSS in the feature space we derive the minimized feature space and then implement the PSO method with dynamic adaptation to also minimize the local optima drawback as discussed above in the previous section.

3.3 Dynamically Adaptive PSO

As mentioned by Xeuming Yang in [11] local optima occurs as the velocity of a particle rapidly approaches zero as it draws closer to a near optimal solution. Thus the convergence rate decreases at later stages of evolution leading to a standstill to the algorithm optimization. According to [11]

Pseudo Code of PSO:

Initial Population

While (number of generations, or stopping criterion is not met)

For p=1 to number of particles

If the fitness of is greater than the fitness of

Then Update =

For k ∈ Neighborhood of

If the fitness of is greater than that of gbest then

Update *gbest* =

Next k

For each dimension d

= w + (-)+ × (—)

If ≠ (,) then

= max (min (,),)

= +

Next d

Next p

Next generation until stopping criterion

<u>Pseudo code for Particle Update</u>

= w + (—) + × (—) (1)

S( ) = ——————— (2)

if (< S( )) then = 1; else = 0 (3)

3.4 Our Proposed Approach

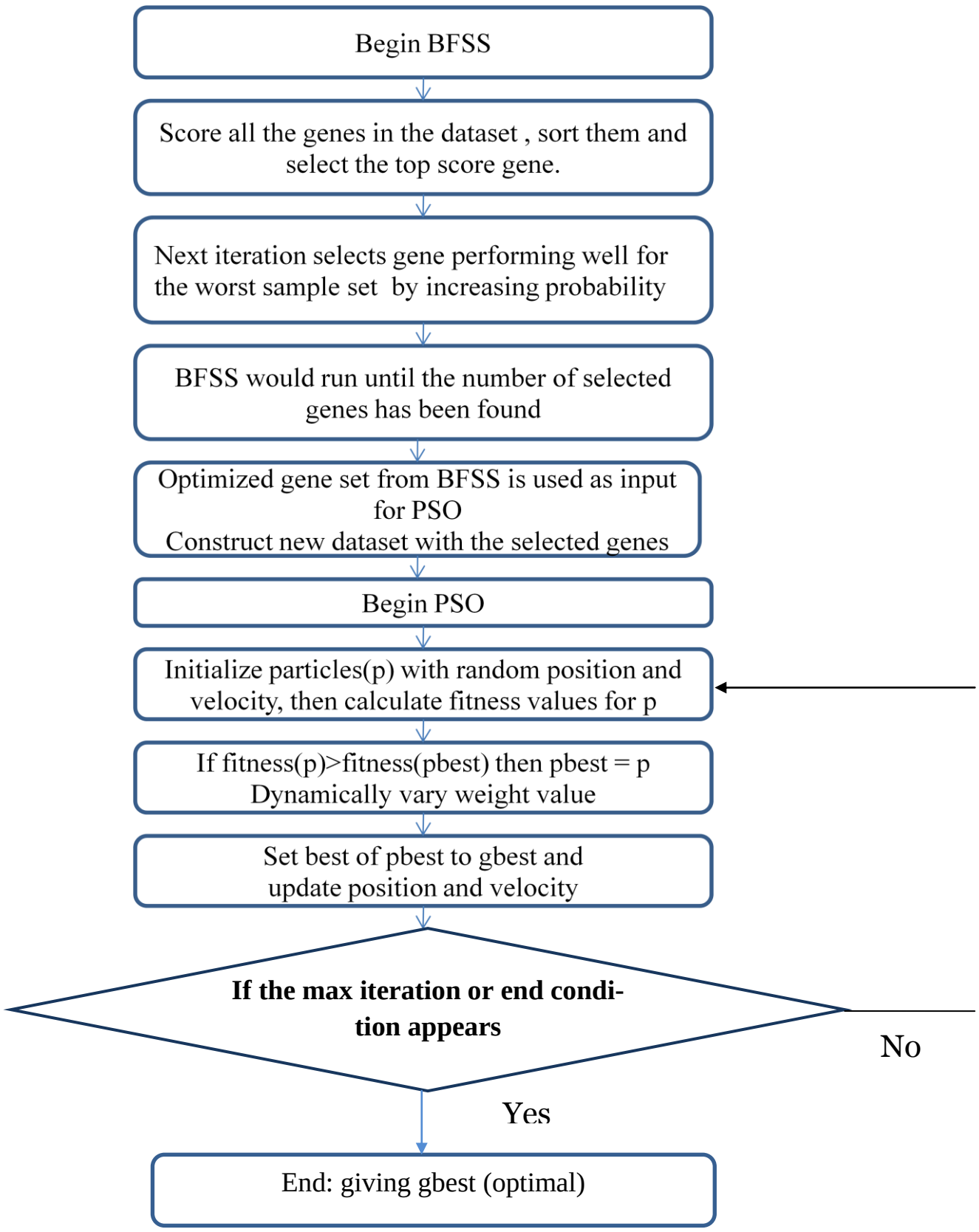


Fig 4: MFS-PSO Flow Diagram

In each iteration the best particle is being chosen according to their fitness values and the number of genes selected is also updated if it has a reduced value.

Chapter 4

Experimental Analysis and Result Comparison

We have used Matlab R2009 as the main platform of our work and used several function references under Bioinformatics Toolbox.

4.1 Dataset Details

These microarray datasets on which we have applied our proposed approach are: Acute Lymphoblastic leukemia cancer (ALL), Lung cancer and colon cancer. Table 1 summarizes the data sets. In the ALL dataset there are 72 tissue samples (47 B-cell and 25 T-cell). In the lung dataset there are 181 tissue samples (47 MPM and 134 ADCA). The training set contains 32 of them, 16 MPM and 16 ADCA. The rest 149 samples are used for testing. Each sample is described by 7130 genes. Colon dataset contains 62 samples collected from colon-cancer patients. Among them, 40 tumor biopsies are from tumors (labeled as "negative") and 22 normal (labeled as "positive") biopsies are from healthy parts of the colons of the same patients. 2000 genes out of around 6500 genes were selected based on the confidence in the measured expression levels.

Table 1: Summery of Microarray datasets.

Dataset	**Number of Classes**	**Number of Samples in the Dataset**	**Number of Genes**
ALL (Leukemia)	2 (B-cell ALL and T-cell ALL)	72 (47 B-cell and 25 T-cell ALL)	7130
LUNG	2 (MPM and ADCA)	181 (47 MPM and 134 ADCA)	12533
COLON	2 (Normal and tumor)	62 (22 normal and 40 tumor)	2000

From Table 1 we can see that dataset for Colon cancer contains the lowest number of genes comparing to other two datasets, exposing higher possibilities of misclassifications and over fitting. It is because the more the number of samples the more we can train classifiers to classify test samples.

4.2 Experimental Settings

Table : Parameters for MFS-PSO.

Dataset	Colon	Leukemia	Lung Cancer
Initial Population	1300	2139	3760
Inertia Weight	0.5	0.3 0.5 0.8 1.1	0.5
Acceleration Factor	0.2	0.2 2	0.2
Mfspso (Acceleration Handler)	10 14	10 14	10 14

U

A

lower value of mfspso increases the probability of selecting a certain gene while a higher value suppresses the acceleration gained by a certain particle depending on the relational logic used to select to deselect a dimension.

The inertia weight **w** controls the impact of the previous velocity of a particle on its current one.Proper adjustment of the inertia weight and the acceleration factors c , c is very important. Too small parameter adjustment would cause too small particle movement and result in useful data, but is time-consuming

While for excessive adjustment particle movement will be excessive, causing the algorithm to weaken early, so that a useful feature set cannot be obtained.

Hence, suitable parameter adjustment enables particle swarm optimization to increase the efficiency of feature selection.

In our approach the best results were obtained with c1 and c2 = 0.2, w = 0.5 and mfspso = 10 for Lung Cancer and Colon datasets and for Leukaemia dataset we tuned the acceleration factors to 2 ,inertia weight to 1.1 and mfspso was retained to 10 to obtain higher accuracies and lowest number of informative genes.

From our experiments we have found that highest classification accuracies were found with the **acceleration handler** = 10. To S inertia weight in and thus the algorithm gets stuck in local optima.

In our approach we have selected a moderate value of weight = 0.5 and compared it to higher values in the experimental analysis which is discussed later.

Local Optima – Convergence rate decreases at later stages of evolution; when closer to a near optimal solution the algorithm stops optimizing and accuracy becomes limited. Velocity of particle rapidly approaches 0.

4.3 Performance Analysis

Our implementation begins with Boost Feature Subset Selection (BFSS) where we select a particular number of genes using T-score method. The main objective to perform BFSS is to provide PSO with a better initial population rather than generating it randomly thus avoiding early convergence and over-fitting problems.

In case of implementing BFSS we have used T-scoring to calculate the scoring for each gene within a sample. T-scoring process is joined with the BFSS algorithm successfully and we got all the scores of the genes for retrieving the best scored gene.

Then we applied BFSS on the three important available microarray datasets and got the reduced number of genes to apply as the input for MFS-PSO. These are shown in the table below-

Table : Reduced number of genes by Boost Feature Subset Selection

Datasets	Original Number Of Genes	BFSS output
Leukemia (ALL) Cancer	7130	2139
Lung Cancer	12533	3760
Colon	2000	1300

We have taken roughly 30% of the existing dataset by applying BFSS for leukemia and lung cancer datasets. For these MFS-PSO will take 2139 and 3760 genes as initial population. As number of genes existing in Colon dataset is only 2000, the output for BFSS was set to 65% of total number of genes, which is 1300 genes.

Table 4: Classification accuracies and number of genes selected by MFS-PSO for leukemia (ALL) dataset

No. of Runs	Leukemia Cancer			
	MFSPSO-KNN	#Selected Genes(KNN)	MFSPSO-SVM	#Selected Genes(SVM)
1	88.89%	7	100%	6
2	81.94%	4	100%	6
3	86.11%	5	100%	7
4	84.72%	5	100%	8
5	83.33%	8	100%	8

6	86.11%	10	100%	6
7	83.33%	7	100%	7
8	95.83%	15	100%	7
9	87.50%	10	100%	4
10	88.89%	11	100%	7
Average ±S.D	87.92±5.98	8.2±2.2	100%±0	6.6±2

Table 4 shows us that the highest accuracy was achieved by MFSPSO-SVM with accuracy 100% and it was achieved by only selecting 4 genes from a total of 7130 genes. On the other hand MFSPSO-KNN has the highest accuracy of 95.83% with the selection of 15 genes.

Table 5: Classification accuracies and number of genes selected by MFS-PSO for lung cancer dataset

No. of Runs	**Lung Cancer**			
	MFSPSO-SVM	**#Selected Genes(SVM)**	**MFSPSO-KNN**	**#Selected Genes(KNN)**
1	97.78%	19	96.13%	22
2	98.89%	20	96.69%	18
3	98.89%	18	97.24%	23
4	95.56%	19	96.13%	12
5	97.33%	15	95.58%	13
6	96.67%	15	93.92%	20
7	91.11%	20	96.13%	24
8	94.44%	25	94.48%	24
9	94.44%	19	94.48%	15
10	95.56%	13	95.03%	20
Average ±S.D	96.07±1.63	18.3±3.3	95.98±1.5	20.1±2.6

Table 5 shows us that the highest accuracy was achieved by MFSPSO-SVM with accuracy 98.89% and it was achieved by selecting 18 genes from a total of 12533 genes. On the other hand MFSPSO-KNN has the highest accuracy of 97.24% with the selection of 23 genes. Here for the both classifiers the standard deviation is 1.63 and 1.5.

Table 6: Classification accuracies and number of genes selected by MFS-PSO for colon cancer dataset

No. of Runs	Colon Cancer			
	MFSPSO-SVM	**#Selected Genes(SVM)**	**MFSPSO-KNN**	**#Selected Genes(KNN)**
1	83.87%	8	78.42%	5
2	87.61%	7	87.10%	8
3	87.10%	7	82.26%	11
4	83.72%	6	77.42%	6
5	83.87%	10	80.65%	8
6	87.10%	8	72.58%	4
7	83.72%	10	70.97%	9
8	87.10%	7	80.65%	5
9	91.61%	7	79.03%	6
10	87.01%	11	80.65%	6
Average ±S.D	86.27±2.8	8.2±4.2	78.97±4.01	6.8±2.8

Table 6 shows us that the highest accuracy was achieved by MFSPSO-SVM with accuracy 91.61% and it was achieved by only selecting 7 genes from a total of 2000 genes. On the other hand MFSPSO-KNN has the highest accuracy of 87.10% with the selection of 8 genes and the standard deviation is 2.8.

Table 7: Classification accuracies and number of genes selected by BPSO-SVM and BFSS for leukemia (ALL) cancer dataset

No. of Runs	Leukemia Cancer			
	BPSO-SVM	**#Selected Genes(BPSO-SVM)**	**BFSS**	**#Selected Genes(BFSS)**
1	94.29	72	80.65%	5
2	91.43	72	84.25%	10
3	88.57	70	92.47%	20
4	91.43	68	92.36%	25
5	91.43	77	90.85%	30
6	91.43	73	91.68%	35
7	94.29	75	93.34%	40
8	91.43	65	94.94%	50
9	91.43	67	93.71%	60
10	91.43	76	94.22%	65
Aver-age ± S.D	**91.72±2.29**	**71.5±4.05**	**91.85±3.09**	**34±5.6**

Table 7 shows us that the highest accuracy was achieved by BPSO-SVM with accuracy 94.29% and it was achieved by selecting 72 genes from a total of 7130 genes. On the other hand BFSS has the highest accuracy of 94.94% with the selection of 50 genes.

Table 8: Classification accuracies and number of genes selected by BPSO-SVM and BFSS for lung cancer dataset

No. of Runs	Lung Cancer			
	BPSO-SVM	**#Selected Genes(BPSO-SVM)**	**BFSS**	**#Selected Genes(BFSS)**
1	94.29%	72	81.11%	5
2	91.43%	72	81.92%	10
3	88.57%	70	82.44%	20
4	91.43%	68	82.86%	25
5	91.43%	77	82.98%	30
6	91.43%	73	82.12%	35
7	94.29%	75	81.76%	40
8	91.43%	65	83.62%	50
9	91.43%	67	83.81%	60
10	91.43%	76	84.44%	65
Aver-age ± S.D	**91.72±2.3**	**71.5±6.06**	**82.71%±1.73**	**34±6.5**

Table 8 shows us that the highest accuracy was achieved by BPSO-SVM with accuracy 94.29% and it was achieved by selecting 75 genes from a total of 12533 genes. On the other hand BFSS has the highest accuracy of 84.44% with the selection of 65 genes.

Table 9: Classification accuracies and number of genes selected by BPSO-SVM and BFSS for colon cancer dataset

No. of Runs	Colon Cancer			
	BPSO-SVM	#Selected Genes(BPSO-SVM)	BFSS	#Selected Genes(BFSS)
1	87.1	398	80.65%	5
2	87.1	407	81.05%	10
3	90.32	176	82.87%	20
4	87.1	183	82.64%	25
5	83.87	222	82.64%	30
6	83.87	75	81.31%	35
7	90.32	397	83.12%	40
8	83.87	360	83.71%	50
9	87.1	414	82.54%	60
10	80.65	396	83.65%	65
Aver-age ± S.D	**86.13±3.8**	**302.8±4.6**	**82.42%±1.73**	**34±5.4**

Table 9 shows us that the highest accuracy was achieved by BPSO-SVM with accuracy 90.32% and it was achieved by selecting 397 genes from a total of 2000 genes. On the other hand BFSS has the highest accuracy of 83.71% with the selection of 50 genes. Here the standard deviation is 1.73.

4.4 Comparative Analysis

Dataset	ACO	PSO	GA-SVM	CGA-SVM	BPSO-SVM	BFSS	MFSPSO-KNN	MFSPSO-SVM
Leukemia	83.89% ------	84.22% ------	83.80±2.14; **88.24%**	91.53±2.48; **96.12%**	91.72±2.29 **94.29%**	91.85±3.09 **94.94%**	87.92±5.98 **95.83%**	100±0 **100%**
Colon	76.87% ------	76.89% ------	83.48±1.77; **86.27%**	86.36±1.46; **89.74%**	86.13±3.8 **90.32%**	82.42±1.73 **83.71%**	78.97±4.01 **87.10%**	86.27±2.8 **91.61%**
Lung	79.76% ------	80.09% ------	84.85±1.54; **87.92%**	86.89±1.54; **89.32%**	91.72±2.3 **94.29%**	82.71±1.73 **84.44%**	95.98±1.5 **97.24%**	96.07±1.63 **98.89%**

Dataset	Original Genes	Selected Genes				
		GA-SVM	CGA-SVM	BPSO-SVM	BFSS	MFSPSO-KNN	MFSPSO-SVM
Leukemia	7130	23±6.06	17.90±1.91	71.5±4.05	34±5.6	8.2±2.2	**6.6±2**
Colon	**2000**	23±6.06	25.5±1.43	302.8±4.6	34±5.4	8.2±4.2	**6.8±2.8**
Lung	**12533**	23±6.06	32.3±2.58	71.5±6.06	34±6.5	20.1±2.6	**18.3±3.3**

From Table 10 and 11 we can get the best accuracies and the number of genes selected. We have taken number of genes as X-axis and Y-axis with both accuracies and ratio for all the independent runs. Here, Ratio = Accuracy / Number of genes selected.

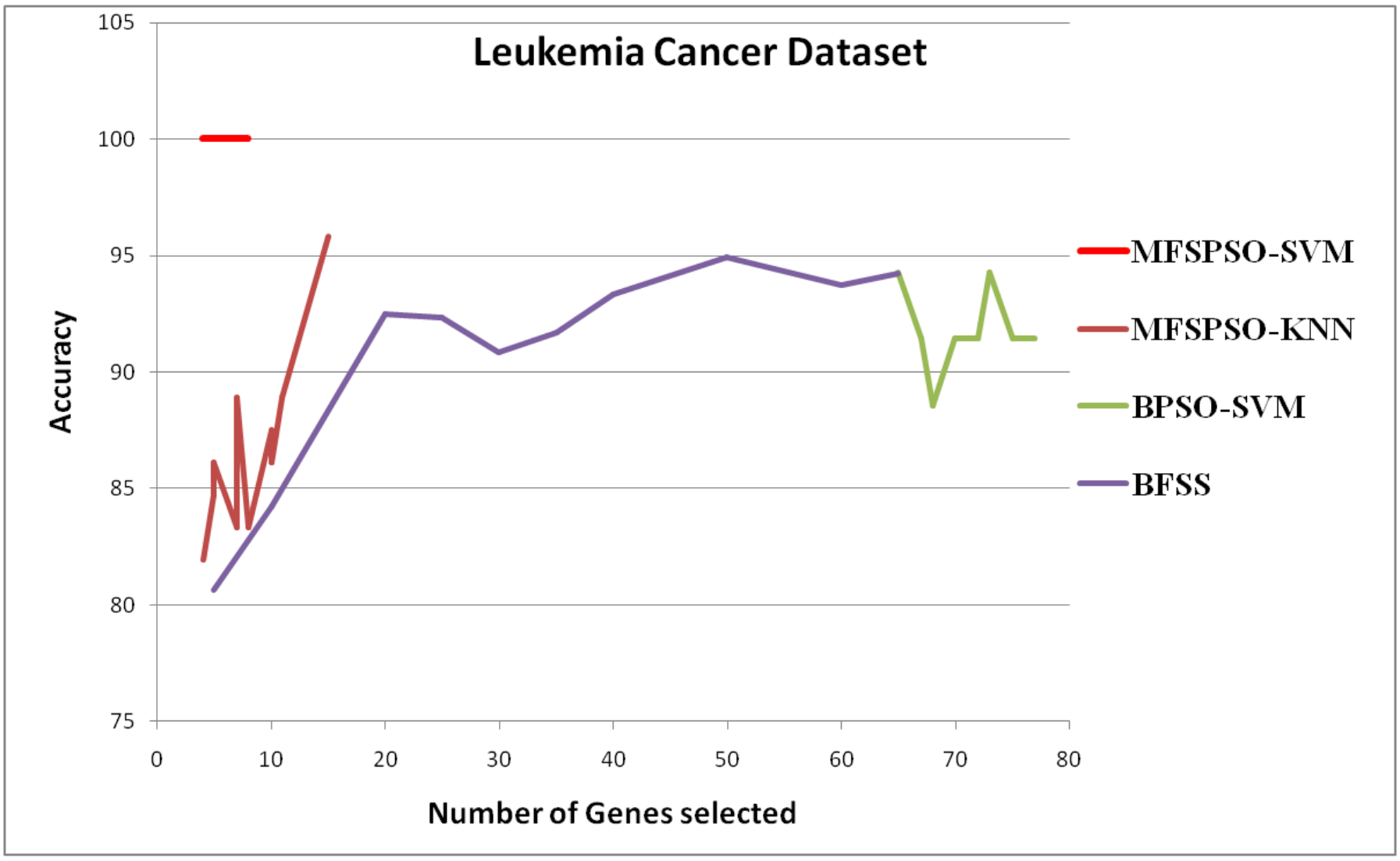


Fig 5: Graphical representation of comparison for leukemia (ALL) cancer dataset.

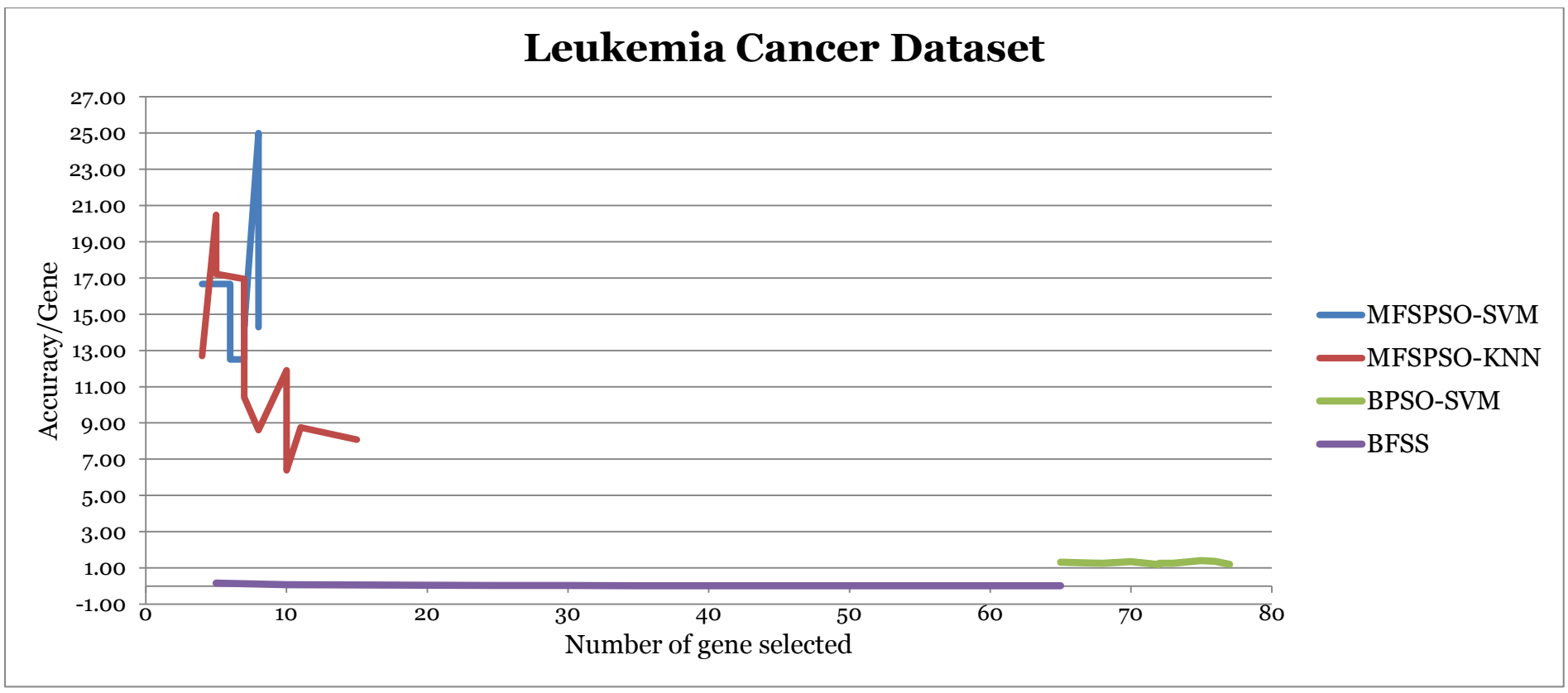


Fig 6: Graphical representation of comparison for leukemia (ALL) cancer dataset.

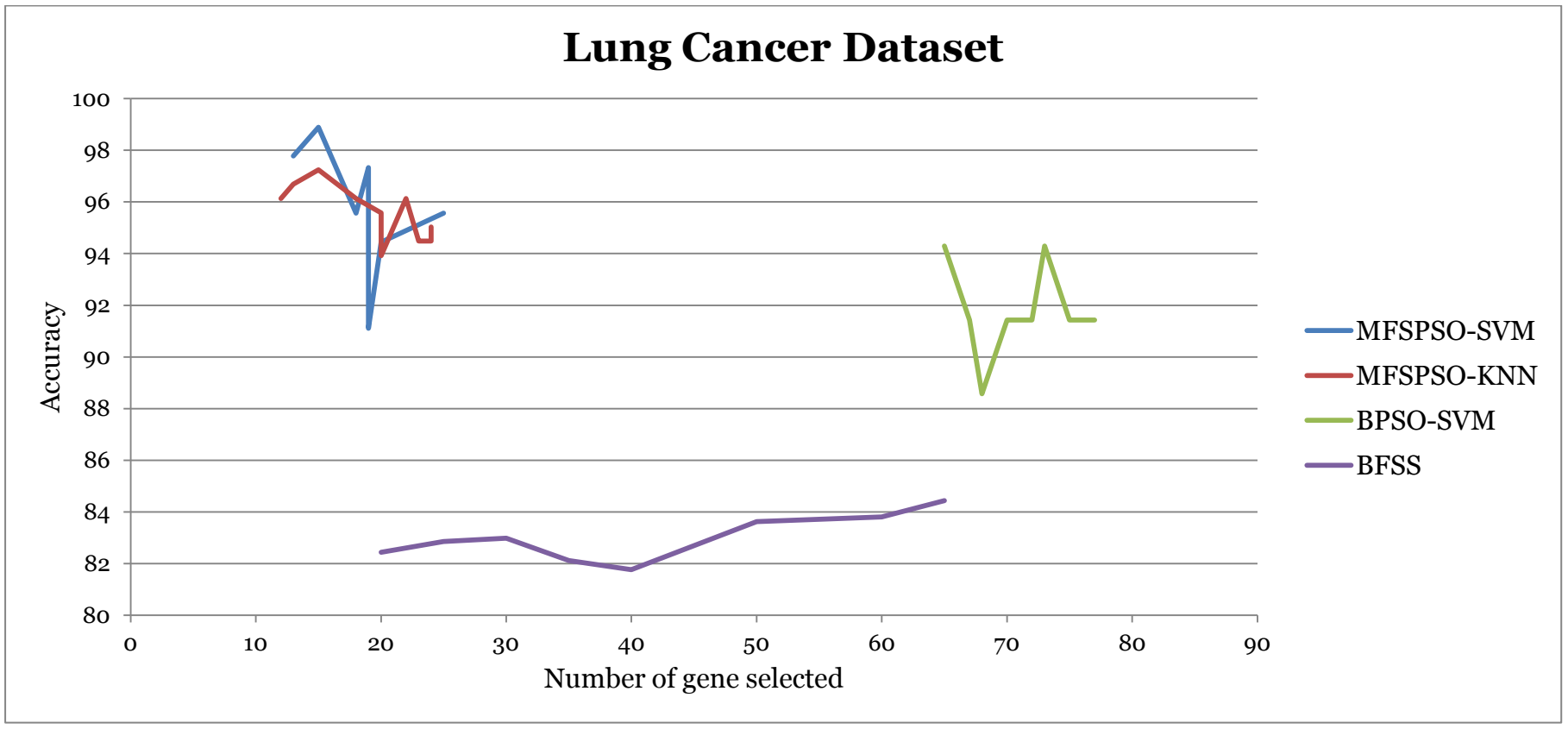


Fig 7: Graphical representation of comparison for lung cancer dataset.

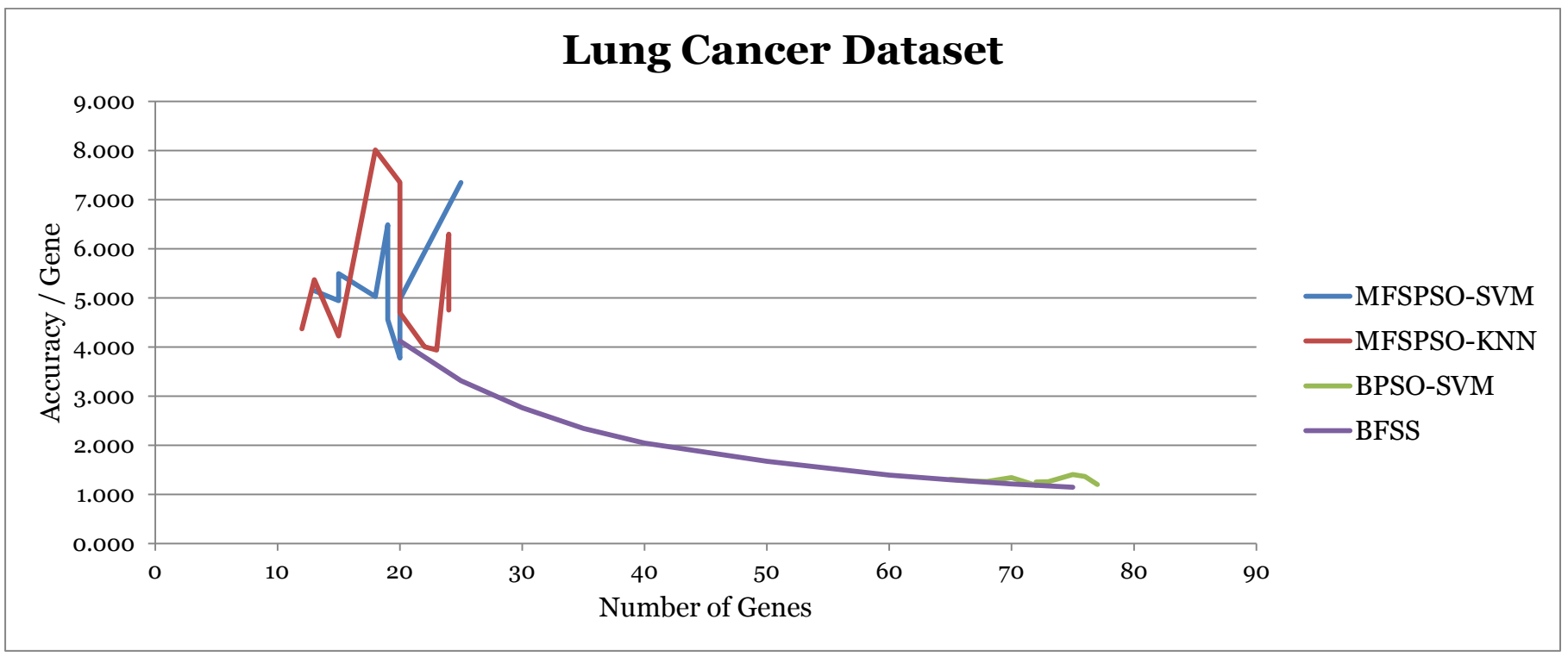


Fig 8: Graphical representation of comparison for lung cancer dataset.

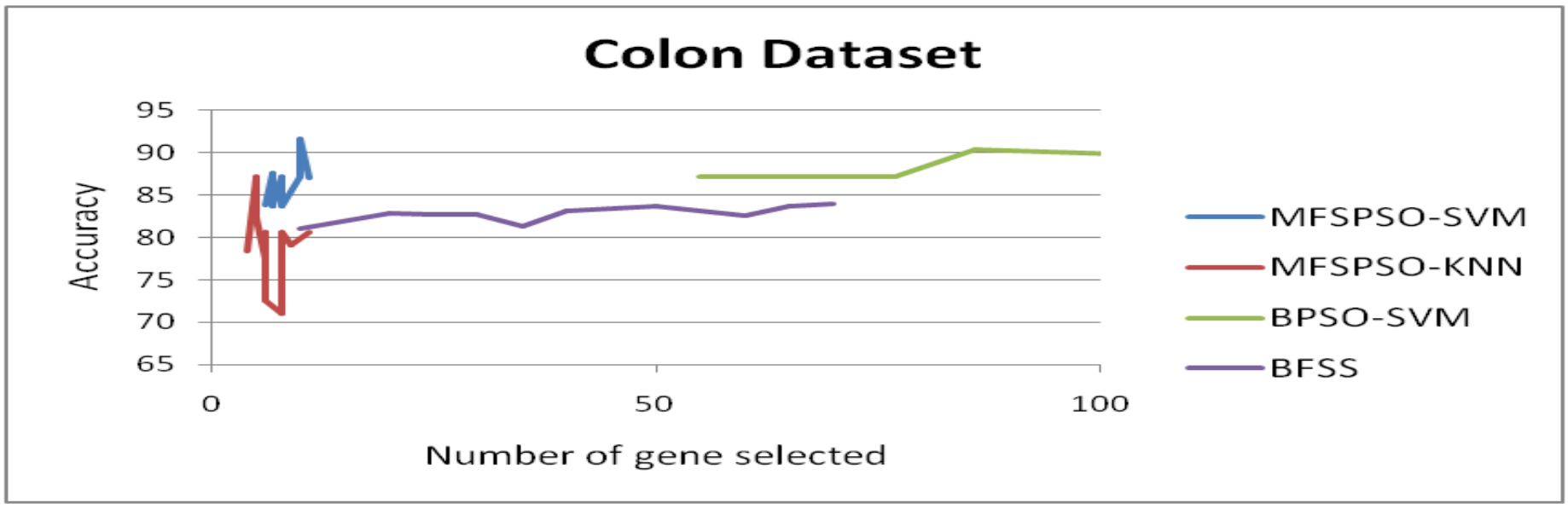


Fig 9: Graphical representation of comparison for colon dataset.

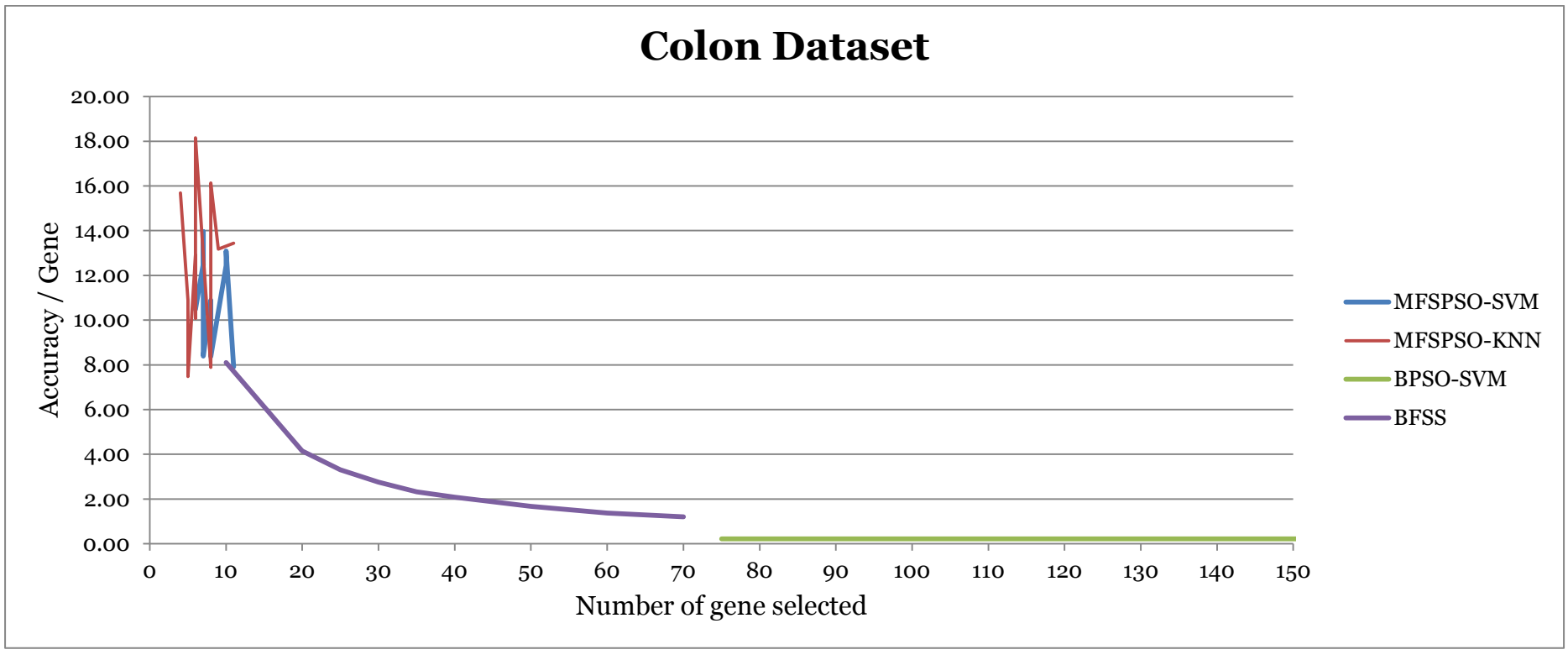


Fig 10: Graphical representation of comparison for colon dataset.

From the above graphs we can see that our approach for either SVM or KNN is giving better results from two different perspectives. In our approach, we got the best accuracies for selecting less number of genes thus giving a better result.

Top most informative genes selected from *Leukemia Dataset*

(7130 genes)

Rank	Index	Accuracy
1	735	100%
2	503	
3	986	
4	1138	

Top most informative genes selected from Colon *Dataset*

(2000 genes)

Rank	Index	Accuracy
1	105	91.61%
2	50	
3	328	
4	592	
5	635	
6	953	
7	982	

Top most informative genes selected from *Lung Cancer Dataset* (12533 genes)

Rank	Index	Accuracy
1	83	
2	1065	
3	1220	
4	1462	
5	1494	
6	1631	
7	1788	
8	1992	
9	2067	98.89%
10	2069	
11	2704	
12	2711	
13	3003	
14	3134	
15	3478	
16	3504	
17	3623	
18	3759	

Here we have shown that the above selected genes for the three datasets, we have got the highest accuracies. Thus these genes are the most informative ones.

Chapter 5

Conclusion

In this study, we have seen that our approach works well on microarray datasets other than existing approaches. Traditional BPSO suffered from the problem of taking input of a large number of genes which will result in the local optima problem. BFSS scores individual gene but does not consider the collective predictive power of genes. Thus we feed the output of BFSS to PSO to maintain correlation. For future development this framework can also be used for other high dimensional data used in other fields such as archeology, geography, climate study, data mining, image processing and many others. The data analysis is expected to produce good results for these other datasets. Even in our field of specialization we have not used our framework on all the microarray datasets such as brain cancer, bone cancer, stomach cancer etc. Also there are many classifiers available; in our study we have used SVM and KNN classifiers. Other classifiers such as C4.5, Naïve Bayes Classifier can also be integrated with our framework for an enhanced comparative analysis.

```
%%%%%%%%%%%%   Dataset 2(ALL)   %%%%%%%%%%%%%%%%
gene = csvread('ALL.csv');
[rows, cols] = size(gene);

Data_X = gene(1:95,2:267);
Data_Y = gene(96:128,2:267);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   BFSS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

bootstrapSize = 2*rows;
worstSampleSetSize = ceil( 0.96 * bootstrapSize );
degradationFactor = 0.96;

toSelectGeneNumber = ceil(0.65*(cols-1));
probability = zeros(rows,2);
probability(:,1) = 1/rows;   %1st column holds the probability of selecting
the sample for generating bootstrap dataset
probability(:,2) = [1:rows]; %2nd column stores the sample indices

G = zeros(1,toSelectGeneNumber);    %selected gene array
E1 = gene; % E1 is the training dataset

geneIndicesOfMainDataset = 1:cols-1;

for gene_select=1:toSelectGeneNumber
    bootstrap_B = [];

    if(gene_select == 1)

        %generate bootstrapSample Set

        bootstrapSampleSetIndices = ceil(rows.*rand(bootstrapSize,1));

        %generate bootstrap B from training set E1
        for i = 1:bootstrapSize
            bootstrap_B(i,:) = E1(bootstrapSampleSetIndices(i,1),:);
        end
        %bootstrap_B(:,1) = []; %eliminate 1st column of labels

        %calculate t-score for each gene
        [PValues, TScores] = mattest(transpose(Data_X),transpose(Data_Y));
        [bestScore bestScoreGeneindex] = max(TScores);  %keep track of best
score

        %add top ranked gene to G
        G(1,gene_select) = bestScoreGeneindex;

    else

        %generate bootstrap B from new training set E1 based on probability
        for boot=1:bootstrapSize
            randSampleIndex =
ceil(degradationFactor*length(sorted_probability).*rand());
            randSample = sorted_probability(randSampleIndex,2) ;
            bootstrapSampleSetIndices(boot,1) = randSample ;
            bootstrap_B(boot,:) = E1(bootstrapSampleSetIndices(boot,1),:);
```

```
        end
        %bootstrap_B(:,1) = []; %eliminate 1st column of labels

        sorted_bootstrap_B = sortrows(bootstrap_B,1);
        [r c] = size(bootstrap_B);
        data_y_index = 1;
        dataX = [];
        dataY = [];

        for k=1:r

            if(sorted_bootstrap_B(k,1)<0)
                dataX(k,:) = sorted_bootstrap_B(k,2:cols-gene_select-1);
            else
                dataY(data_y_index,:) = sorted_bootstrap_B(k,2:cols-
gene_select-1);
                data_y_index = data_y_index + 1 ;
            end
        end
        [pvalues, tscores] = mattest(transpose(dataX),transpose(dataY));
        [bestScore bestScoreGeneindex] = max(tscores);  %keep track of best
score

        %convert bestScoreGeneindex into geneIndexofMainDataset
        G(1,gene_select) = geneIndicesOfMainDataset(1,bestScoreGeneindex);

    end

     for i = 1:length(bootstrapSampleSetIndices)


        %generate new sample set by subtracting 1 sample from
bootstrapSampleSet in each loop
        temp = bootstrapSampleSetIndices;
        S1 = temp;
        S1(i,:) = []; %S1 is the new sample set

        %which sample is eliminated?->keeping track of that sample
        eliminated_sample(i,1) = bootstrapSampleSetIndices(i,1);

        %%% score the new set %%%
        %generate bootstrap_C dataset for remaining samples
        for j = 1:length(S1)
            bootstrap_C(j,:) = [E1(S1(j,1),1)
E1(S1(j,1),bestScoreGeneindex)];
        end

        %split the bootstrap_C into Data_X and Data_Y for Tscoring
        sorted_bootstrap_C = sortrows(bootstrap_C,1);
        [r c] = size(bootstrap_C);
        data_y_index = 1;

        for k=1:r

            if(sorted_bootstrap_C(k,1)<0)
                num(k,1)= sorted_bootstrap_C(k,2);
            else
                num2(data_y_index,1) = sorted_bootstrap_C(k,2);
                data_y_index = data_y_index + 1 ;
```

```
            end

        end

        %find tscore for best gene selected
        [pvalues, tscore] = mattest(transpose(num),transpose(num2));
        worstSampleScore_S0(i,1) = tscore;
        worstSampleScore_eliminatedSample(i,[1 2]) = [tscore eliminat-
ed_sample(i,1)];

    end

    %sort worstSampleScore_S0
    sorted_S0 = sort(worstSampleScore_S0,'descend');
    sorted_S0_eliminatedSample = sortrows(worstSampleScore_eliminatedSample,-
1); % sort in descending order % This matrix contains..
    % both the score and the worst_sample_index

    %%% Add top worst samples to set S1 %%%
    S1 = sorted_S0_eliminatedSample( 1:worstSampleSetSize,2 ) ;    %S1 is the
worst sample set
    S2 = sorted_S0_eliminatedSample( worstSampleSetSize +
1:length(sorted_S0_eliminatedSample), 2 ); %S2 is the best set of sample

    %%% reduce probability of best set of samples %%%
    S2_unique = unique(S2);
    [ro co] = size(S2_unique);
    for x=1:ro
        probability(S2_unique(x,1),1) = (degradationFactor * probabil-
ity(S2_unique(x,1),1));
    end

    %normalising
    probability(:,1) = probability(:,1)/sum(probability(:,1)) ;
    sorted_probability = sortrows(probability,-1);

    %Remove g from E1 and
    E1(:,bestScoreGeneindex) = [];

    %update geneIndicesOfMainDataset
    geneIndicesOfMainDataset(:,bestScoreGeneindex) = [];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   new DATASET from BFSS
as input for PSO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%new DATASET from BFSS as input for PSO

gene = [ gene(1:rows,1) gene(1:rows,G(1,1:toSelectGeneNumber)) ];
[rows, cols] = size(gene);
P = zeros(10, cols-1, 'uint32'); % P is the set of chromosomes
B = rand(10,cols-1);
geneselected = zeros(10,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%    PSO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%randomly generate particles in array P
for i=1:10
    %geneselected(i,1) = count;
```

```
    count = 0;
    for j=1:cols-1
        %          if(count==20)
        %              break;
        %          end;
        if(B(i,j)>.554 && B(i,j)<.559)
            P(i,j)=1; count = count + 1;
        end;
    end;
end;


%generate one - zero species matrix to denote true class

col=zeros(rows,1);
%flag = 0;
species = zeros(rows,1);  %generate one - zero species matrix to denote true
class
for s = 1:rows
    if(gene(s,1)>0)
        species(s,1)=1;
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%generate array of selected gene indices:
%initial fitness for each chromosome


numGene =ones(10,1);
for chromosome=1:10
    num = 1;
    for j=1:cols-1
        if(P(chromosome,j)==1)

            %              select(chromosome,numGene(chromosome,1)) = j + 1;
            %              numGene(chromosome,1) = numGene(chromosome,1) + 1;
            select(chromosome,num) = j + 1;
            num = num+ 1;
        end
    end

    %initial fitness for each chromosome

    data = gene( : , select(chromosome,[1:num-1]) );
    [train, test] = crossvalind('holdOut',species);
    cp = classperf(species);                          %initializes the CP - CP
holds classifier information( accuracy )

    svmStruct = svmtrain(data(train,:),species(train));
    class = svmclassify(svmStruct,data(test,:));

    classperf(cp,class,test);
    fitness(chromosome,1) = cp.CorrectRate;


end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate fitness of updated chromosome
pbest = zeros(10,1);
pbest_chromosome=zeros(10,cols-1);
gbest = 0;
gbest_chromosome = zeros(1,cols-1);

c1 = 2;
c2 = 2;
w = 0.9;
Vmax = 2;
Vmin = 0;
v = zeros(10,cols-1);
S_VnewPD = zeros(10,cols-1);


for generation=1:5
    select(:,:) = [];  %new
    for chromosome=1:10
        %update pbest
        if(fitness(chromosome,1)>pbest(chromosome,1))
            pbest(chromosome,1) = fitness(chromosome,1);
            pbest_chromosome(chromosome,:) = P(chromosome,:);
        end

        %update gbest
        for neighbour=1:10
            if(pbest(neighbour,1)>gbest)
                gbest = pbest(neighbour,1);
                gbest_chromosome(1,:) = P(neighbour,:);
                gbest_index = neighbour;
            end
        end

        num = 1;
        for dimension=1:cols-1

            %update velocity
            v(chromosome,dimension) = (  w * v(chromosome,dimension) ) + ( c1
* rand() * (pbest(chromosome,1) - P(chromosome,dimension)) ) + ( c2 * rand()
* (gbest - P(chromosome,dimension)) );

        %update position
        ex = exp(- v(chromosome,dimension));
        S_VnewPD(chromosome,dimension) = 1/(1 + ex);

        r=0.7;
        if(r<S_VnewPD(chromosome,dimension))
            P(chromosome,dimension) = 0;
            %update select array

        else
            P(chromosome,dimension) = 1;
            select(chromosome,num) = dimension + 1;
            num = num + 1;
        end

end

% find updated chromosome fitness
```

```
        data = gene( : , select(chromosome,[1:num-1]) );
        [train, test] = crossvalind('holdOut',species);
        cp = classperf(species);                        %initializes the CP -
CP holds classifier information( accuracy )

        svmStruct = svmtrain(data(train,:),species(train));
        class = svmclassify(svmStruct,data(test,:));

        classperf(cp,class,test);
        fitness(chromosome,1) = cp.CorrectRate;

    end


end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%number of gene selected in the best chromosome
numberOfGeneInBestParticle=0;
for i=1:cols-1
    if(gbest_chromosome(1,i)==1)
        numberOfGeneInBestParticle = numberOfGeneInBestParticle+1;
    end
end
```

References

1. **Li-ye-chuang et al**.(2012) A Hybrid BPSO-CGA Approach for Gene Selection and Classification of Microarray *Data JOURNAL OF COMPUTATIONAL BIOLOGY Volume , Number ,*
2. **Bing Xue et.al**.(2012)Single Feature Ranking and Binary Particle Swarm Optimisation based Feature Subset Ranking for Feature Selection. *In Thirty-Fifth Australasian Computer Science Conference*
3. **Rahmat Allah Hooshmand et.al**. (2012)Fuzzy Optimal Phase Balancing of Radial and Meshed Distribution Networks using BF-PSO Algorithm. *In IEEE , VOL. 27, NO. 1*
4. **Enrique Alba**(2011) Parallel multi-swarm optimizer for gene selection in DNA microarrays *Springer Science+Business Media, LLC*
5. **Mohd Saberi Mohamad et.al** (2011) A Modified Binary Particle Swarm Optimization for selecting the Small Subset of Informative Genes from Gene Expression Data. *In IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, VOL. 15, NO. 6.*
6. **Yuanning Liu et al.** (2011) An Improved Particle Swarm Optimization for Feature Selection
7. **Ruichu Cai et al.**(2011) A new hybrid method for gene selection
8. **Wei Zhao et al.**(2011) A Novel Framework for Gene Selection
9. **Kelly Fleetwood et al.**(2010) An Introduction to Differential Evolution
10. **Wei-Neng Chen et al.**(2010) A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems
11. **L.-Y. Chuang et al.**(2010) Correlation-based Gene Selection and Classification Using Taguchi-BPSO
12. **Mohd Saberi Mohamad et al.(2009)** Particle swarm optimization for gene selection in classifying cancer classes. *In Artif Life Robotics (2009) 14:16–19*
13. **Sheng Ding-**(2009) Feature Selection based F-score and ACO Algorithmin Support Vector Machine
14. **Shutao Li et al.**(2008) Gene selection using hybrid particle swarm optimization and genetic algorithm
15. **Mojtaba Ahmadieh Khanesar et al.**(2007) A Novel Binary Particle Swarm Optimization. *In 15th Mediterranean Conference on Control and Automation ,Athens,Greece.*
16. **Xueming Yang et al.** (2007) A modified particle swarm optimizer with dynamic adaptation
17. **R. K. Agrawal, et al.**(2007) A Hybrid Approach for Selection of Relevant Features for Microarray Datasets
18. **Qi Shen et al.**(2006) A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification
19. **Chung-Jui Tu et al.**(2006) Feature Selection using PSO-SVM

20. **Xian Xu and Aidong Zhang** (2006) Boost Feature Subset Selection: A New Gene Selection Algorithm for Microarray Dataset
21. **Caruana, Rich and de Sa, Virginia R.** (2003), "Benefitting from the Variables that Variable Selection Discards", *Journal of Machine Learning Research (JMLR), Vol. 3.*
22. **Yuhui Shi et al.**(2001) Fuzzy Adaptive Particle Swarm Optimization.
23. **Daniel W. Dyer** (2008) Evolutionary Computation in Java, *Internet Edition Tech Press*
24. **James Kennedy** (1995) Particle Swarm Optimization