

**ADAPTIVE PARTICLE SWARM  
OPTIMIZATION FOR FEATURE SELECTION ON  
HIGH-DIMENSIONAL DATA**

**A PROJECT REPORT**

*Submitted by*

**CHELAMPALEM SAMPREETH 2017103007**

**GAJARAJAN V Y 2017103012**

**REKKAREKULA ROMEO 2017103617**

*in partial fulfillment for the award of the degree  
of*

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**



**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY:: CHENNAI 600 025**

**MAY 2021**

# **ANNA UNIVERSITY:: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this project report titled “**ADAPTIVE PARTICLE SWARM OPTIMIZATION FOR FEATURE SELECTION ON HIGH-DIMENSIONAL DATA**” is the bonafide work of “**CHELAMPALEM SAMPREETH (2017103007), GAJARAJAN V Y (2017103012) and REKKAREKULA ROMEYO (2017103617)**” who carried out the project work under my supervision, for the fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering.

**Place:** Chennai

**Date:**



**Mr. G. Manikandan**

SUPERVISOR

Teaching Fellow

Department of Computer Science and Engineering

Anna University, Chennai – 25

**COUNTERSIGNED**

**DR. S. Valli**

**HEAD OF THE DEPARTMENT**

Department of Computer Science and Engineering

Anna University, Chennai – 600 025

## ABSTRACT

Feature selection has been a powerful tool to handle high-dimensional data. FS helps in improving the accuracy and interpretability of the learnt models, shortening the learning time, and reducing the storage space of the dataset, eliminating irrelevant and redundant features. However most existing feature selection methods are still prone to stagnation in local optima or/and high computation cost. A global search technique is needed to explore this huge search space better. PSO (Particle Swarm Optimization) is a population-based algorithm which is well-known with global search ability that has been successfully applied to classification problems in many domains.

JLOPSO I (Jump Local Optima Particle Swarm Optimization I) and JLOPSO II (Jump Local Optima Particle Swarm Optimization II) are completely redesigned to develop a local search protocol to avoid premature convergence problem which is to jump from local optima to global optima for better solutions, and to introduce a concept of splitting a high-dimensional problem of feature selection into a number of low-dimensional sub-problems obviously improves the scalability of high-dimensional data handling by PSO, to develop an adaptive sub-swarm size adjustment method for removing redundant particles automatically or inserting new particles of good diversity from/into a sub-swarm for maintaining an acceptable size for each sub-swarm and enabling particles to have different and shorter lengths, which defines smaller search space and therefore, improves the performance of PSO. Comparing the performance of the incorporating algorithms with the existing methods.

## திட்டப்பணிசுருக்கம்

அம்சத் தேர்வு உயர் பரிமாண தரவைக் கையாள ஒரு சக்திவாய்ந்த கருவியாகும். கற்ற மாதிரிகளின் துல்லியம் மற்றும் விளக்கத்தை மேம்படுத்துவதற்கும், கற்றல் நேரத்தை குறைப்பதற்கும், தரவுத்தொகுப்பின் சேமிப்பிட இடத்தைக் குறைப்பதற்கும், பொருத்தமற்ற மற்றும் தேவையற்ற அம்சங்களை அகற்றுவதற்கும் எஃப். எஸ் உதவுகிறது. இருப்பினும் தற்போதுள்ள பெரும்பாலான அம்சத் தேர்வு முறைகள் உள்ளூர் ஆப்டிமா அல்லது / மற்றும் அதிக கணக்கீட்டு செலவில் தேக்க நிலைக்கு ஆளாகின்றன. இந்த பெரிய தேடல் இடத்தை சிறப்பாக ஆராய உலகளாவிய தேடல் நுட்பம் தேவை. பி.எஸ்.ஓ (துகள் திரள் உகப்பாக்கம்) என்பது மக்கள் தொகை அடிப்படையிலான வழிமுறையாகும், இது உலகளாவிய தேடல் திறனுடன் நன்கு அறியப்பட்டதாகும், இது பல களங்களில் வகைப்படுத்தல் சிக்கல்களுக்கு வெற்றிகரமாக பயன்படுத்தப்படுகிறது.

ஜே. எல். ஓ. பி. எஸ். ஓ - I (உள்ளூர் ஆப்டிமா துகள் திரள் உகப்பாக்கம் I) மற்றும் ஜே. எல். ஓ. பி. எஸ். ஓ II (ஜம்ப் லோக்கல் ஆப்டிமா துகள் திரள் உகப்பாக்கம் II) ஆகியவை உள்ளூர் தேடல் நெறிமுறையை உருவாக்க முற்றிலும் மறுவடிவமைப்பு செய்யப்பட்டுள்ளன., மற்றும் அம்சத் தேர்வின் உயர் பரிமாண சிக்கலை பல குறைந்த பரி-மாண துணை சிக்கல்களாகப் பிரிக்கும் கருத்தை அறிமுகப்படுத்துவது, பி. எஸ். ஓ ஆல் உயர் பரிமாண தரவு கையாளுதலின் அளவை மேம்படுத்துகிறது, அதற்கான தகவமைப்பு துணை திரள் அளவு சரிசெய்தல் முறையை உருவாக்க. தேவையற்ற துகள்களை தானாக நீக்குதல் அல்லது நல்ல துணை வேறுபாட்டின் புதிய துகள்களை ஒவ்வொரு துணை திரட்டிற்கும் ஏற்றுக்கொள்ளக்கூடிய அளவைப் பராமரிப்பதற்கும், துகள்கள் வெவ்வேறு மற்றும் குறுகிய நீளங்களைக் கொண்டிருப்பதற்கும் ஒரு சிறிய திரைக்குள் செருகுவது, இது சிறிய தேடல் இடத்தை வரையறுக்கிறது, எனவே, செயல்திறனை மேம்படுத்துகிறது பி.எஸ்.ஓ. இணைக்கும் வழிமுறைகளின் செயல்திறனை ஏற்கனவே உள்ள முறைகளுடன் ஒப்பிடுதல்.

## ACKNOWLEDGEMENT

First and foremost, we would like to express our deep sense of gratitude to our project guide, **Mr. G. Manikandan**, Teaching Fellow, Department of Computer Science and Engineering, Anna University, for his excellent guidance, counsel, continuous support and patience. He has helped us to come up with this topic and guided us in the development of this project. We thank him for his kind support and for providing necessary facilities to carry out the work.

We are thankful to the project committee members **Dr. S. Bose**, Professor, Department of Computer Science and Engineering, Anna University and **Dr. V. Mary Anita Rajam**, Professor, Department of Computer Science and Engineering, Anna University for their valuable guidance and support.

We extend our gratitude to **Dr. S. Valli**, Head of the Department, Department of Computer Science and Engineering, Anna University, for providing a conducive environment and facilities for the project.

We express our thanks to all the faculty members and non-teaching staff or having helped us in one way or other for the successful completion of the project.

We thank our parents and friends for their indirect contribution to the successful completion of our project.

**CHELAMPALEM SAMPREETH GAJARAJAN V Y REKKAREKULA ROMEYO**

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT .....</b>	iii
	<b>TAMIL ABSTRACT .....</b>	iv
	<b>LIST OF TABLE .....</b>	viii
	<b>LIST OF FIGURES .....</b>	ix
	<b>LIST OF ABBREVIATIONS .....</b>	x
<b>1.</b>	<b>INTRODUCTION .....</b>	<b>1</b>
	1.1 FEATURE SELECTION .....	1
	1.2 PROBLEM STATEMENT .....	3
	1.3 OBJECTIVES OF THE PROJECT .....	3
	1.4 OUTLINE OF THE REPORT .....	4
<b>2.</b>	<b>LITERATURE REVIEW .....</b>	<b>5</b>
<b>3.</b>	<b>PROPOSED SYSTEM ARCHITECTURE .....</b>	<b>11</b>
	3.1 ARCHITECTURE DIAGRAM .....	11
	3.2 JUMP LOCAL OPTIMA PARTICLE SWARM OPTIMIZATION (JLOPSO I) .....	12
	3.2.1 Space Division - Feature Importance .....	13
	3.2.2 Initialize Sub – Swarm sizes .....	14
	3.2.3 Evaluate Fitness .....	14
	3.2.4 Adaptive Adjustment Sub-Swarm Size .....	15
	3.2.5 Initialize sub-swarm .....	16
	3.2.6 Construct Solution .....	16

3.2.7 Local search I . . . . .	17
3.3 JUMP LOCAL OPTIMA PARTICLE SWARM	
OPTIMIZATION II (JLPSO II) . . . . .	18
3.3.1 Rearrange features -Feature ranking . . . . .	19
3.3.2 Population Division . . . . .	20
3.3.3 Learning probability and exemplar Assignment . . . . .	21
3.3.4 Evolutionary process . . . . .	21
3.3.5 Local Search II . . . . .	21
<b>4. IMPLEMENTATION AND RESULT ANALYSIS . . . . .</b>	<b>23</b>
4.1 DATASET DESCRIPTION . . . . .	23
4.2 METRICS FOR EVALUTION . . . . .	23
4.3 JUMP LOCAL OPTIMA PARTICLE SWARM	
OPTIMIZATION (JLOPSO I) . . . . .	23
4.4 JUMP LOCAL OPTIMA PARTICLE SWARM	
OPTIMIZATION II (JLPSO II) . . . . .	25
4.5 RESULTS . . . . .	26
<b>5. CONCLUSION AND FUTURE WORK . . . . .</b>	<b>38</b>
5.1 CONCLUSION . . . . .	38
5.2 FUTURE WORK . . . . .	38
<b>REFERENCES . . . . .</b>	<b>39</b>

## **LIST OF TABLES**

- 4.1 Table showing the description about each data set
- 4.2 Table showing the number features selected for each dataset by JLOPSO I
- 4.3 Table showing the number features selected for each dataset by different algorithms
- 4.4 Table showing the number features selected for each dataset by JLOPSO II
- 4.5 Table showing the number features selected for each dataset by different algorithms



## **LIST OF FIGURES**

### 3.1 Architecture Diagram

#### 4.1 JLOPSO I execution for Lymphoma Dataset

#### 4.2 JLOPSO I execution for Colon Dataset

#### 4.3 JLPSO II execution for Lymphoma Dataset

#### 4.4 JLPSO II execution for LSVT Dataset

#### 4.5 Accuracy Graph for Different M Values for Lymphoma dataset

#### 4.6 Comparing the accuracies for Lymphoma Dataset

#### 4.7 Accuracy Graph for Different M Values for LSVT dataset

#### 4.8 Comparing the accuracies for LSVT Dataset

#### 4.9 Accuracy Graph for Different M Values for COLON dataset

#### 4.10 Comparing the accuracies for Colon Dataset

#### 4.11 Accuracy Graph for Different K Values for LYMPHOMA dataset

#### 4.12 Comparing the accuracies for Lymphoma Dataset

#### 4.13 Accuracy Graph for Different K Values for LSVT dataset

#### 4.14 Comparing the accuracies for LSVT Dataset

#### 4.15 Accuracy Graph for Different K Values for COLON dataset

#### 4.16 Comparing the accuracies for Colon Dataset

## **LIST OF ABBREVIATIONS**

<b>FS</b>	Feature Selection
<b>PSO</b>	Particle Swarm Optimization
<b>JLOPSO I</b>	Jump Local Optima Particle Swarm Optimization I
<b>JLOPSO II</b>	Jump Local Optima Particle Swarm Optimization II
<b>EC</b>	Evolutionary computation
<b>APSO</b>	Accelerated Particle Swarm Optimization
<b>PPSO</b>	Potential Particle Swarm Optimization
<b>BBPSO</b>	Barebones Particle Swarm Optimization
<b>VS-CCPSO</b>	Variable Size – Cooperative Coevolutionary Particle Swarm Optimization
<b>VLPSO-LS</b>	Variable Length Particle Swarm Optimization – Local Search
<b>SU</b>	Symmetrical Uncertainty
<b>GA</b>	Genetic Algorithm
<b>HHO</b>	Harris Hawks Optimization

# **CHAPTER 1**

## **INTRODUCTION**

This chapter deals with the detailed introduction about feature selection and importance.

Due to the advent of technology, data sources have been increased for example sensing data, internet data, trading data etc. In real-world, we have to deal with big data which is associated with three key concepts: variety, volume, velocity. where the data can have very large number of features when compared with number of instances called as high dimensional dataset, however not all the features are important to build an algorithm e.g., classification algorithm since many of features may be redundant and irrelevant, which drastically reduces the performance of the model and also increases the time taken to build the model. To overcome the problems mentioned, feature selection gives a good hand, to remove the redundant and irrelevant features from the whole feature set.

### **1.1 FEATURE SLECTION**

High-dimensional dataset size can be reduced in two ways i.e., sample set reduction and feature set reduction. we will concentrate on feature set reduction, further there are two types in feature set reduction namely feature extraction [19]-[20] and feature selection. The major difference is feature extraction has transformation of original feature set to novel feature set, whereas feature selection has no transformation but selects subset of features from the existing feature set. Feature set reduction have to deal with two key concepts: feature relevance and feature redundancy. whereas feature redundant deals with multiple features(multivariate) but feature relevance deals with individual feature(univariant).

A feature can be classified into four main types: strongly relevant, weakly relevant but not redundant, irrelevant, redundant features [15,16]. The motive behind feature selection is increasing relevance and decreasing redundancy.

There will be  $2^m-1$  feature subsets for  $m$  features [21,22], so to find the global optimum solution from the search space is considered to be a challenging task. All the feature selection algorithm is mainly having four basic steps in finding the feature subset: subset generation, subset evaluation, stopping criteria, validation of the results [17]. Categories of feature selection methods: filter, wrapper, embedded, hybrid, and some additional methods (such as structural dependency related and streaming oriented), the first four methods mentioned deal with feature independency or near-independency [18].

Glimpse of the various feature selection methods, first filter methods pick out features on the basis of a performance metric regardless of the employed data modelling algorithm, secondly wrapper methods makes use of the quality of the performance on a modelling algorithm, which is taken as black box evaluator for the feature subsets, thirdly embedded methods performs feature selection during the data modelling algorithm execution, then hybrid method which combines the best properties of filter and wrapper method by combination of filter and wrapper methods.

However most existing feature selection methods are still prone to stagnation in local optima or/and high computation cost. FS is a combinatorial optimization problem with huge search space to explore with these a global search technique is needed. Evolutionary computation (EC) techniques are well-known for their global search potential. Particle swarm optimization (PSO) is a population-based evolutionary algorithm which is well-known for its global search ability/potential. It is attracted from the social behavior of birds flocking or fish school. PSO works as follows it maintains a swarm of particle, each of which represents the candidate solution to the problem. These particles communicate with each other about their best-found solution so far as individual and team best solution, then these particles can move towards the most interesting areas in the search space to explore and exploit better solution.

PSO has been mainly used to solve single-objective, unconstrained optimization problems, but nowadays PSO has been developed to solve constrained and multi-objective problems. But PSO is still limited to low-dimensional dataset, its performance on high-dimensional dataset for feature selection is not so good due to such as fixed-length representation of particles, which requires significant amount of memory and computation cost when applying for feature selection on high-dimensional dataset and no divide and conquer strategy while dealing with high-dimensional dataset, premature convergence of the swarm. The mentioned limitation motivates us to develop a PSO such that it can overcome the limitation for finding the global optima candidate solution. we are going to propose variable length representation of particles to overcome the fixed-length representation problem, divide the given high-dimensional feature selection problem into many low dimensional PSO to improve the scalability, automatic adding and deletion of particles to the swarm depending on the situation demand such that to reduce the computation burden whenever needed and also incorporate local search strategy to overcome the premature convergence problem.

## **1.2 PROBLEM STATEMENT**

PSO performance on high-dimensional data is still limited due to FS methods use the fix-length representation, which requires a significant amount of memory and computation time. The premature convergence, which is a common problem of PSO, especially on high-dimensional data.

## **1.3 OBJECTIVES OF THE PROJECT**

The main objectives of the project are to design particles that can interact smoothly with each other at different lengths for reducing the computational time, to develop an algorithm incorporating a local search protocol to avoid

premature convergence problem, to introduce a concept of splitting a high-dimensional problem of feature selection into a number of low-dimensional sub-problems obviously improves the scalability of high-dimensional data handling by PSO, to develop an adaptive sub-swarm size adjustment method for removing redundant particles automatically or inserting new particles of good diversity from/into a sub-swarm for maintaining an acceptable size for each sub-swarm and Comparing the performance of the incorporating algorithms with the existing methods.

## **1.4 OUTLINE OF THE REPORT**

Chapter 2 discusses the literature survey which are the articles and the papers referred for the idea. Chapter 3 talks about the actual system design and how each and every part of the module is built with the underlying algorithms.

## CHAPTER 2

### LITERATURE REVIEW

Simon Fong et.al [3] proposed a novel lightweight feature selection to deal with high-dimensionality and streaming format of data feeds in Big Data. By using accelerated particle swarm optimization (APSO) type of swarm search, a feature selection was designed specially to mining streaming data on the fly within reasonable processing time. For performance evaluation of the proposed algorithm, a list of high-dimensionality Big Data are put under the test.

To tackle the problem of continuous generation of fresh data at all times, an incremental computation approach is required which is able to monitor large scales of data dynamically. To achieve robustness, high accuracy and minimum pre-processing latency, lightweight incremental algorithms should be used. Five datasets of different domains that have a very large amount of features, from UCI archive are used to compare traditional classification model induction and their counterpart in incremental inductions. The evaluation results show that the incremental method obtained a higher gain in accuracy per second incurred in the pre-processing.

Yong Zhang et.al [4] introduced a concept of multi-objective particle swarm optimization (PSO) for cost-based feature selection problems, the scenario where user is not only interested in maximizing the performance of classification but also minimizing the cost that may be associated with features is called as cost-based feature selection. Generate pareto front of nondominated solutions, that is, feature subsets, to meet different criteria of decision-makers in real-world application is the task of this paper.

A probability-based encoding technology and an effective hybrid operator, together with the ideas of the crowding distance, the external archive, and the Pareto domination relationship, are applied to PSO, to enhance the search capability of the proposed algorithm (HMPSOFS). The proposed

algorithm is compared with three multi-objective feature selection algorithms on five benchmark datasets. The experimental results show that proposed algorithm can search the solution space more effectively to obtain a set of non-dominated solutions instead of a single solution. The drawback of the proposed algorithm is it is unknown whether the achieved pareto fronts can be improved or not. But HMPSOFS can able to give a set of good feature subsets. In the future, the author will further investigate the multi-objective PSO-based feature selection approach to better explore the Pareto front of non-dominated solutions in various feature selection problems.

Binh Tran et.al [5] developed a concept called potential particle swarm optimization (PPSO) to reduce the search space of the problem and also better evaluation function to steer the search. PPSO make two-stage approach of discretization and feature selection (FS) into single stage using barebones particle swarm optimization (BBPSO). PPSO uses new PSO representation to synchronously identifying cut-points for discretizing multiple features and selecting features. The experimental result on ten-high dimensional datasets (microarray datasets) exhibits that less than 5% of total features was selected by PPSO. PPSO achieves higher accuracy than two-stage approach on seven datasets, where two-stage approach uses BBPSO for FS on already discretized datasets. Drawback of PPSO is it uses a binary discretization method, which may not work well on data that wants to be discretized into multiple intervals. The author said that they will work on the mentioned problem to optimize the search process.

Luca Bravi et.al [6] developed a multivariate feature ranking method, in order to examine the relevance of features a concave approximation of zero-norm function is employed and defined a smooth, global optimization problem. The system is based on the solution of instances of the global optimization problem depending on handy training data. The system consists of building an analytical model of the process using handy data and also solutions from



multiple global optimization problems. Both the real and artificial datasets had been used to evaluate the system, and pointed out that the proposed method of feature is a valid alternative to the existing method in terms of effectiveness. The limitation of the system is high CPU time, to overcome this limit the parallelization techniques will be used to get solution of multiple global optimization problems.

Zhi-Zhong Liu et.al [7] presented a framework named as adaptively tune the coordinate systems (ACoS) in nature-inspired optimization algorithms (NIOAs). To tackle the performance problem of many NIOAs which strongly depend on their implemented coordinate system, Eigen coordinate system (ECS) is established in ACoS. ECS is developed by making use of cumulative population distribution information, which can be acquired based on covariance matrix adaptation strategy and additional archiving mechanism. From the combination of Eigen coordinate system and original coordinate system, one is selected based on the probability vector. ACoS has been applied to two typical paradigms of NIOAs namely particle swarm optimization (PSO) and differential evolution (DE), over 30 test functions with 30Dimensions and 50Dimensions at the 2014 IEEE Congress on Evolutionary Computation. Computational result shows that the proposed system is effective when compared with some other Eigen coordinate system-based methods.

Kamlesh Mistry et.al [8] proposed feature optimization technique using evolutionary particle swarm optimization for facial expression recognition. At first discriminative initial facial representation is generated, then the proposed system a PSO variant embedded with micro genetic algorithm (mGA) is used to perform feature optimization. mGA involves some of techniques namely cooperation of global exploration and local exploitation search mechanism to deal to premature convergence problem of conventional PSO. To recognize seven facial expressions multiple classifiers are used. The study uses within-domain images from extended Cohn Kanade and also cross-domain images from

MMI benchmark databases. In comparison with other PSO variants, classical GA and conventional PSO, the proposed system outperforms in facial image recognition.

Makoto Yamada et.al [9] introduced the concept of novel Hilbert-Schmidt Independence Criterion Lasso (HSIC Lasso) to handle ultra-high-dimensional data. The system promises to identify optimal subset of features with minimum redundancy and maximally predictive capacity. The experiment is carried out on classification of phenotypes in human prostate cancer patients and to identify the presence of enzymes among protein structure. The proposed method named as Least Angle Nonlinear Distributed (LAND) feature selection performs better on three real-world, high dimensional datasets (P53, PC, Enzyme). It also achieved higher accuracy with the dimensionality reduction of 99.998 percent on ultra-high-dimensional data.

Narges Armanfard et.al [10] proposed a novel localized feature selection (LFS) approach which allows the feature set to adjust to local variations in the sample space and also a method to measure the similarity between query datum to each of the representative classes. The system is independent to distribution of data over sample space because it makes no assumption on underlying structure of sample. The proposed method is robust against the overfitting problems. Experiment on eleven synthetic and real-world data sets shows that the proposed algorithm performs well when compared to previous state-of-the-art feature selection algorithms. In some cases the proposed localized feature selection method outperforms global feature selection methods.

Bin Hu et.al [11] developed an improved shuffled frog leaping algorithm, which explores the possible feature subsets to identify the feature subset that minimizes the irrelevant features and maximizes the predictive accuracy in high-dimensional biomedical data. The proposed algorithm introduces the concept of chaos memory weight factor, an absolute balance group strategy and also an adaptive transfer factor. The proposed method is compared with particle swarm

optimization, genetic algorithms and shuffled frog leaping algorithm which point out that proposed system outperforms the other algorithms in the identification of relevant subset of features and classification accuracy.

Jorge González-López et.al [12] developed a distributed model for mutual information (MI) adaptation on scalar features and multiple labels using Apache Spark. Based on maximization of mutual information (MI), relevance and minimization of redundancy two approaches are proposed. The former concentrates on selecting feature subset with maximum of MI between features and the labels, whereas the latter deal with minimizes the redundancy between the features. The experiment setup compares the proposed method with other methods for distributed feature selection for multilabel data on 10 dataset with 12 metrics shows that proposed algorithm outperforms and also reduces the CPU runtime in order of magnitude.

Mohammed A. Ambusaidi et.al [13] introduced a mutual information based algorithm to select the optimal features analytically for classification. The linearly and Non linearly dependent data features are handled by this mutual information based feature selection algorithm called Flexible Mutual information Feature Selection (FMIFS) algorithm. Using FMIFS algorithm features are selected to build an Intrusion Detection System (IDS), named Least Square Support Vector Machine based IDS(LSSVM-IDS). The three intrusion detection evaluation datasets, namely NSL-KDD, Kyoto 2006+ and KDD Cup 99 dataset are used to evaluate the performance of LSSVM-IDS. By comparison with the state-of-the-art methods, the evaluation result shows that FMIFS algorithm identifies more vital features for LSSVM-IDS to achieve lower computational cost and better accuracy.

Myeongsu Kang et.al [14] proposed an outlier insensitive hybrid feature selection (OIHFS) methodology to reduce diagnostic performance deterioration caused by outliers in data-driven diagnostics. The dataset is classified into two different sub datasets named an analysis dataset and an evaluation dataset are

used. The most discriminatory feature subset for bearing fault diagnosis is determined by using the analysis dataset and the evaluation dataset is used for efficacy verification of the OIHFS methodology. The developed approach achieved diagnostic performance improvements of up to 30.0% over the conventional method in terms of the average of  $N_{iterations} \times k$  classification accuracies (ACA).

Rami Sihwail et.al [23] developed an Improved Harris Hawks Optimization (IHHO) using the elite opposite-based learning (EOBL) technique and proposing a new search mechanism called three search strategies (TSS) search for feature selection. A twenty benchmark datasets are collected from the UCI repository and the scikit-feature project, which represent different levels of feature dimensionality, such as low, moderate, and high. The experimental results have confirmed the dominance of IHHO over the other optimization algorithms such as Generic algorithm (GA), Grasshopper Optimization Algorithm (GOA), Particle Swarm Optimization (PSO), Ant Lion Optimizer (ALO), Whale Optimization Algorithm (WOA), Butterfly Optimization Algorithm (BOA) and Slime Mould Algorithm (SMA) in different aspects, along with accuracy, fitness fee, and feature selection.

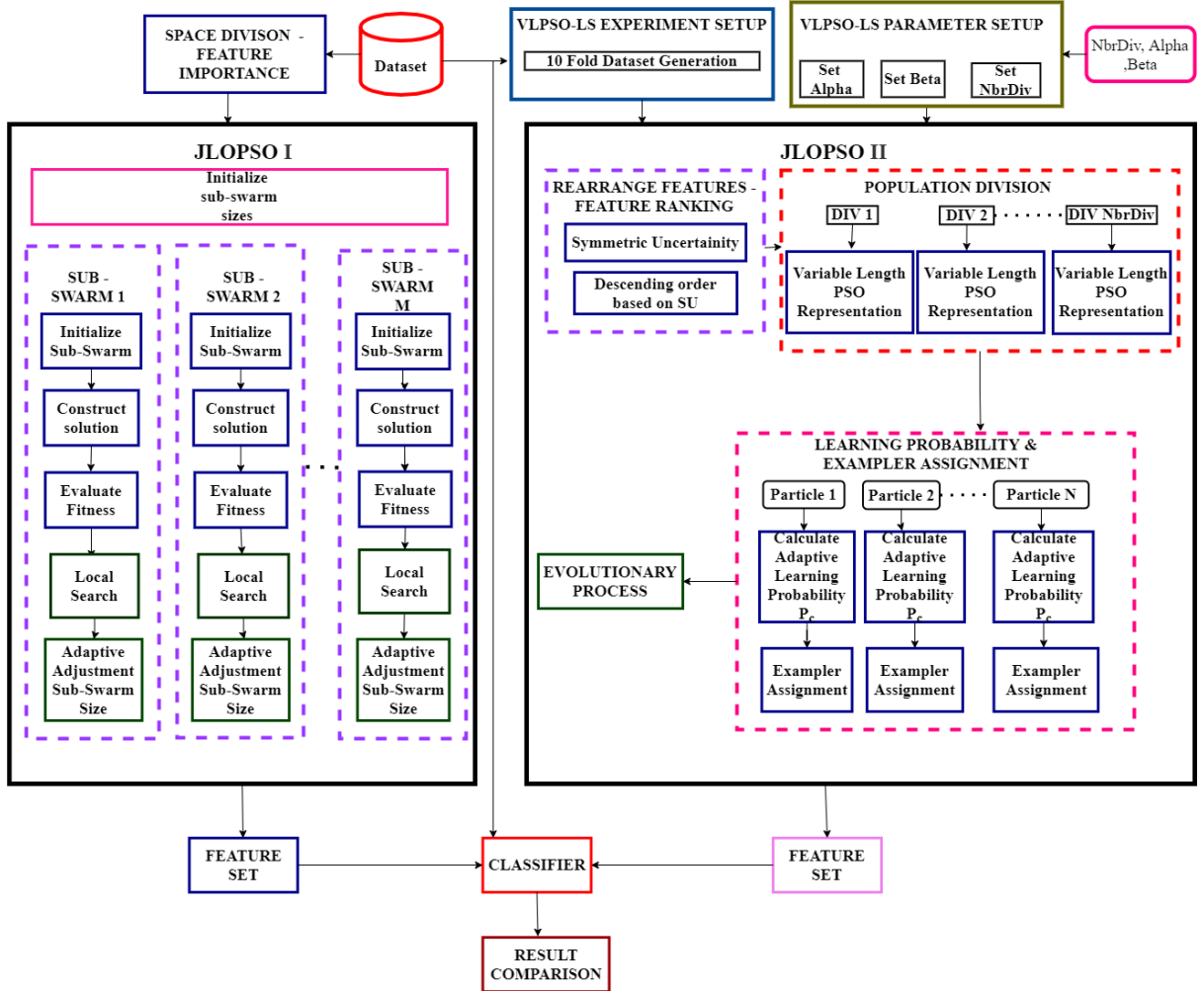
## CHAPTER 3

### PROPOSED SYSTEM ARCHITECTURE

This chapter describes the overall system design and detailed explanation about each of the modules and the corresponding underlying algorithms.

#### 3.1 ARCHITECTURE DIAGRAM

The overall architecture is divided into two approaches namely the VS-CCPSO approach and the VLPSO-LS approach. For the VS-CCPSO approach the dataset is given as input directly to the Space division-feature importance method to get the corresponding feature set. The dataset is also given to the VLPSO-LS method to get the corresponding feature set and then both results are compared.



**Figure 3.1** Architecture Diagram

### 3.2 JUMP LOCAL OPTIMA PARTICLE SWARM OPTIMIZATION I (JLPSO I)

For this algorithm the dataset is given as input. Firstly we calculate the Symmetrical Uncertainty  $SU$  between all features and class. From these calculations the importance of each feature is analyzed and the weak or irrelevant features are removed. After deleting the irrelevant features, the remaining features are sorted in descending order based on the Symmetrical Uncertainty  $SU(F, C)$  values of each feature. Then these sorted features are uniformly divided into  $M - subspaces$ . Then  $Fim$  (Feature Importance of a subspace) for each sub-spaces is calculated. After that initial size of all sub-spaces (i.e., Number of particles for a sub-space) is calculated. Update the size of sub-spaces such that they are within bounds. Initialize all the particles for each sub-swarm. Then for each sub-swarm, the following steps are performed. For each particle in each sub-swarm, construct the solution and calculate the fitness of the particle. If there is a need for Local Search strategy then it is invoked and the Adjustment of the sub-swarm size is made if required and finally the selected feature set is returned.

**Algorithm:**

**Input:** A Dataset.

**Output:** Selected feature set.

**Begin:**

**Step 1:** Calculate the  $SU$  between all features and class.

**Step 2:** Delete the weak or irrelevant features.

**Step 3:** Sort remaining features based on  $SU(F, C)$  in descending order.

**Step 4:** Uniformly divide feature into  $M - subspaces$ .

**Step 5:** Calculate  $Fim$  for each sub-spaces.

**Step 6:** Calculate initial size of all sub-spaces (i.e., Number of particles for a sub-space)

**Step 7:** Update the size of sub-spaces such that they are within bounds.

**Step 8:** For each sub-swarm, initialize all the particles.

**Step 9:** For each sub-swarm, do the following Steps, until stopping criteria is met:

**Step 9.1:** For each particle, construct the solution and calculate fitness of the particle.

**Step 9.2:** Local search strategy invoked if required.

**Step 9.3:** Adjust the sub-swarm size if required.

**Step 10:** Return the selected feature set.

**End**

### 3.2.1 Space Division - Feature Importance

In space division – Feature Importance, the original feature set  $F$ , the class labels  $C$ , and the number of subspaces  $M$ , are given as an input. The importance of each feature is evaluated by using the Symmetrical Uncertainty ( $SU$ ), which tells how much the feature is correlated to the class label. The importance of each feature is recorded and the irrelevant features are removed based on the condition  $SUF(f_n) < \delta$ , where  $\delta = 0.1 * SU_{MAX}$ . Then calculate the size of each feature subspace by using equation (1.1). Sort all the remaining features in  $F$ , in terms of their importance in descending order. These sorted features from 1 to  $l$  get filled into the first feature subspace  $F_1$ , and those from  $l + 1$  to  $2l$  get filled into the second feature subspace  $F_2$ , and so on, this process continues until all features are filled into corresponding subspaces. All these feature subspaces are returned as output.

$$l = \lceil |F|/M \rceil \quad (1.1)$$

### 3.2.2 Initialize Sub – Swarm sizes

For this module, the feature subspace  $F_1 \cup F_2 \cup, \dots, F_M$  is given as input. Calculate  $FIM$ (feature importance of a subspace) for each subspace, by using equation (2.1) and record the importance of each feature subspace.

$$FIM(F_i) = \sum_{K=1}^{|F_i|} SU(F'_K, C) \quad (2.1)$$

where  $|F_i|$  represents the number of features contained in the subspace  $F_i$ ,  $C$  is the set of class labels. Then calculate the initial size for each subspace by using the equation (2.2).

$$S_m = \left\lceil \frac{FIM(F_m)}{\sum_{K=1}^M FIM(F_K)} * N \right\rceil \quad (2.2)$$

where  $N$  is the total number of particles. To reduce the number of particles, we set upper bound and lower bound for all subspace sizes using below expressions (2.3).

$$SN_m = \begin{cases} S_{max}, & S_m > S_{max} \\ S_m, & S_{min} \leq S_m \leq S_{max} \\ S_{min}, & S_m < S_{min} \end{cases} \quad (2.3)$$

where  $S_{max} = \min\left(N, \frac{2N}{M}\right)$ ,  $S_{min} = \min\left(S, \frac{N}{2M}\right)$ , where  $N$  and  $M$  are total number of particles and number of feature subspaces respectively.

### 3.2.3 Evaluate Fitness

In this module, the current particle  $X_{cp}$ , in the feature subspace  $m$ ,  $Gbest$  of all sub-swarms, class label  $C$ . Original feature  $F$  and feature subspace  $m$  are given as input. The complete solution (particle) will be constructed using elite combination strategy. The complete solution is decoded into a feature subset in the following ways. Firstly, we select the  $i^{th}$  feature  $f'_i$  in the sub-swarm  $m$ , if  $X_{cp} = (X'_1, X'_2, \dots, X'_{|F|})$ ;  $X'_i = 1$ , where  $|F|$  is number of features in the sub-swarm  $m$ . The original feature of  $f'_i$  is extracted and then added these to selected feature set( $FS$ ) where it is initially assigned as null. Then extract the data of the



selected features  $X_{FS}$  and class label  $C$ . Then obtain the mean of test-accuracy ( $fit_{X_{cp}}$ ) which are obtained from KNN Leave-One-Out Cross Validation ( $LOOCV$ ) using  $(X_{FS}) \cup C$ . The mean of test-accuracy ( $fit_{X_{cp}}$ ) is returned as output.

### 3.2.4 Adaptive Adjustment Sub-Swarm Size

The particles of the whole swarm are given as input to this module. The relative Convergence ( $rel\_con$ ) for each Sub-Swarm is calculated using equation (4.1). The relative Divergence ( $rel\_Div$ ) for each Sub-Swarm is calculated using equation (4.2) and then these are stored. And then for each Sub-Swarm, check whether sizes of the Sub-Swarm need to be adjusted or not. If the sizes of the Sub-Swarm need not be adjusted then end the whole evolutionary process and output the selected features as of now. If the sizes of the sub-swarm need to be adjusted then calculate the number of particles to be added or deleted from Sun-Swarm. The particle deletion strategy is followed, if the deletion of particles in a Sub-Swarm is needed. Otherwise follow particle generation Strategy and add the particles to the Sub-Swarm.

$$rel\_con_m^t = \frac{[Fbest_m^t - Fbest_m^{t-1}] - [Fbest_m^{t-1} - Fbest_m^{t-2}]}{Z} \quad (4.1)$$

where, best is the fitness of the best particle.

$$rel\_div_m^t = \frac{1}{Nm} \sum_{k=1}^{Nm} |F_m^t(X_i) - Favg_m^t| - \delta \times \frac{1}{Nm} \sum_{i=1}^{Nm} |F_m^{t-1}(X_i) - Favg_m^{t-1}| \quad (4.2)$$

where  $0 < \delta < 1$ .

### 3.2.5 Initialize sub-swarm

For this the sub-swarm size is given as input. The Sub-Swarm has Size  $S$ . For each particle  $i$ , from 1 to  $S$ , we initialize the particle position with a Uniformly Distributed random vector. Then initialize the particle's best known position with its initial position, and calculate the fitness of the particle using the fitness function as  $f(P_i)$ . Then compare fitness of the particle's best known position with the fitness of swarm's best known position and based on this update the swarm's best known position. The particle's of the sub-swarm is returned as output.

### 3.2.6 Construct Solution

Particles of the sub-swarm are given as input for this module. For each particle  $i$  of the sub-swarm, calculate the disturbance factor  $\Delta$ , by using equation (6.1). Then calculate  $|\delta_i|$  of the particle  $i$ , in the current sub-swarm by using equation (6.2) and calculate  $\mu_i$  of the particle  $i$ , in the same sub-swarm using equation (6.3). Then update each dimension value of the particle  $i$  using the expression (6.4). Calculate the fitness of the current-position of the particle  $f(x_i)$  and compare it with the fitness of the best known position of the particle  $f(P_i)$ , if fitness of current-position of the particle  $f(x_i)$  is greater than the fitness of the best known position of the particle  $f(P_i)$  then update  $P_i$  with  $X_i$  otherwise not. And again compare fitness of current-position of particle  $f(x_i)$  with the fitness of the global best of the sub-swarm  $f(g)$ , if fitness of current-position of the particle  $f(x_i)$  is greater than the fitness of the global best of the sub-swarm  $f(g)$  then update the global best  $g_{best}$  with  $X_i$  otherwise the old  $g_{best}$  is maintained. The updated particles are returned as output.

$$\Delta = rand \times |pb_p^t - pb_q^t| \times \exp(f(gb^t) - f(X_i^t)) \quad (6.1)$$

Where  $pb_p^t, pb_q^t$  are randomly selected particle best known position ever.

$$|\delta_i| = |pb_i^t - gb| + \Delta \quad (6.2)$$

$$\mu_i = 0.5(pb_i^t + gb^t) \quad (6.3)$$

$$X_{ij}^{t+1} = \begin{cases} N(\mu_{ij}, |\delta_{ij}|), & rand < prob \\ Pb_{ij}^t, & \text{otherwise} \end{cases} \quad (6.4)$$

Where  $j$  is the dimension of the  $i^{th}$  particle  $rand$  is random number  $[0,1]$   $prob$  is 0.7.

### 3.2.7 Local search I

Global best of the swarm is given as input for this Local Search Strategy. Begin this process by initializing  $i$  with 0 and go for 100 iterations. In each iteration generate a random number from  $[0.85, 1)$  and use it as set lower bound ( $lb$ ). The Symmetrical uncertainty of selected features (i.e.,  $x_f > lb$ , where  $x_f$  is particle value at dimension  $f$ ) are stored in  $SF\_SU$ . Then  $per$  is assigned with a random number from  $[0, 1)$  and calculate  $per$  percentile of  $SF\_SU$  and store in  $limit\_SU$  and assign a random value in  $prob$  vector. The original Global best ( $gbest$ ) is assigned to the temporary global best ( $Tgbest$ ). For each dimension  $j = 1, \dots, D$ , where  $D$  is the number of features, in global best ( $gbest$ ), select a random number from  $[0, 1)$  and store it in  $rand$ . Now the condition  $rand < prob_j$  and  $SU(F_j, C) > limit\_SU$  follows then  $Tgbest_j$  is updated with 1 otherwise another condition  $rand < prob_j$  and  $SU(F_j, C) \leq limit\_SU$  is checked if it follows then  $Tgbest_j$  is updated with 0. The fitness of  $Tgbest$  and number of selected features ( $FS$ ) is evaluated and stored in  $Tgbest\_fitness$  and  $n\_FS1$  respectively. Also the fitness of  $gbest$  and number of selected features ( $FS$ ) is evaluated and stored in  $gbest\_fitness$  and  $n\_FS2$

respectively. Now the following condition is checked ( $Tgbest\_fitness > gbest\_fitness$ ) or ( $Tgbest\_fitness \geq gbest\_fitness$  and  $n\_FS2 > n\_FS1$ ), if it satisfies then we update original global best ( $gbest$ ) with temporary global best ( $Tgbest$ ) otherwise the original global best ( $gbest$ ) is maintained. And at last the global best ( $gbest$ ) is returned as output.

### 3.3 JUMP LOCAL OPTIMA PARTICLE SWARM OPTIMIZATION II (JLPSO II)

A dataset,  $NbrDiv, \alpha, \beta$  are taken as input. Firstly calculate the Symmetrical Uncertainty  $SU$  between all features and class and rearrange the features in descending order of  $SU(F, C)$  where  $F$  and  $C$  are feature and class respectively. Initialize all particles in each division and calculate learning probability  $P_c$  for each particle. Assign exemplar using  $P_c$  for each particle. The following steps to be carried out until the required condition satisfies. For each particle assign exemplar if needed then update the particle velocity and position. Update the fitness and particles ever best known positions ( $pbest$ ) and update fitness and population ever best known positions ( $gbest$ ). If  $gbest$  is not improved for  $\beta$  times, then call the length changing procedure. If check whether particles exemplar assignment required, if true then calculate learning probability  $P_c$  for the particle. If there is a need for Local Search Strategy then it is invoked. And at last the selected feature set is returned as output.

**Algorithm:**

**Input:** A Dataset  $NbrDiv, \alpha, \beta$ .

**Output:** Selected feature set.

**Begin:**

**Step 1:** Calculate the  $SU$  between all features and class.

**Step 2:** Rearrange features in descending order of  $SU(F, C)$ ;

Where  $F$  is feature &  $C$  is class.

**Step 3:** Initialize all particles in each division.

**Step 4:** Calculate learning probability  $P_c$  for each particle.

**Step 5:** Assign exemplars for each particle.

**Step 6:** Do the following steps, until stopping criteria is met:

**Step 7:** For each particle, do the following steps:

**Step 7.1:** Assign exemplar if required.

**Step 7.2:** Update the particle velocity & position.

**Step 7.3:** Update fitness and  $pbest$ ; Where  $pbest$  is particles ever best known positions.

**Step 7.4:** Update fitness and  $gbest$ ; Where  $gbest$  is the population ever best known positions.

**Step 7.5:** If  $gbest$  is not improved for  $\beta$  times, call the length changing procedure.

**Step 7.6:** If check whether particles exemplar assignment required, if true then calculate learning probability  $P_c$  for the particle.

**Step 8:** Local search strategy invoked if required.

**Step 9:** Return the selected feature set.

**End**

### 3.3.1 Rearrange features -Feature ranking

In this module, the original feature set  $F$  and class labels  $C$  given as the input and evaluates the importance of each feature by calculating the Symmetrical Uncertainty using equation (1.1) between each feature and class label. The importance of each feature is recorded and sorted the features according to their importance in the descending order. The sorted feature set is given as output.

$$SU(F, C) = \left[ \frac{IG(F/C)}{H(F)+H(C)} \right] \quad (1.1)$$

$$IG(F/C) = H(F) - H(F/C)$$

### 3.3.2 Population Division

For this module, we give *NbrDiv* as input i.e., the number of divisions that we want to make so that the large search space is divided into smaller subspaces. Initially, we set the *Div* as 1, where *Div* is the current division and perform some required operations until the current division doesn't limit the *NbrDiv*. In each operation, Firstly we calculate the *DivSize*(i.e., size of each division) by using equation (2.1). *DivSize* is the number of particles in a division. Secondly, we calculate the *Parlen*(i.e., length of each particle in current division). *Parlen* is calculated using the equation (2.2). After calculating the *Parlen*, the same *Parlen* is initialized to each particle in current division. After that update the *fitness* and *pbest* for the particle. If we done with current division, then we move to the next division by incrementing *Div* by 1 and this process continues until it doesn't cross the limit if *NbrDiv* and we output the population *P*.

$$DivSize = \frac{PopSize}{NbrDiv} \quad (2.1)$$

Where the *PopSize* is the total number of particles.

$$ParLen = MaxLen * \frac{V}{NbrDiv} \quad (2.2)$$

Where *MaxLen* is the dimensionality of the problem, *V* is the current division.

### 3.3.3 Learning probability and exemplar Assignment

In this, we give population  $P$  as input, which we got it as output from population division. For each particle, in the population  $P$ , we calculate the learning probability  $P_c$ . Based on this  $P_c$  we assign exemplar for the particle. If the exemplar for the particle is assigned, then we set *renewExmpl* for the particle to false and we return the learning probabilities of the population  $P$ .

### 3.3.4 Evolutionary process

In this process we get population  $P$  as input, for each particle  $p$  in population  $P$ , we check the status of *renewExmpl*, if it is true, then exemplar assignment procedure is called, and then *renewExmpl* of particle  $p$  is set to false. After that update the velocity and position of the particle  $p$  and update *fitness* and *pbest* of the particle  $p$ . If the *pbest* is not improved for  $\alpha$  times, then *renewExmpl* of  $p$  is set to true. And then we update *gbest*, if *gbest* is not improved for  $\beta$  times then we call length changing mechanism. Again, if the status of *renewExmpl* is true, then learning probability for all the particles in the population  $P$  is calculated. After that, if there is  $C$  need for local search mechanism, then we call that procedure. Finally we output the selected feature set.

### 3.3.5 Local Search II

The main goal of this Local Search strategy is to find better solutions surroundings the newly found *pbest* by randomly removing some redundant features and adding more relevant features. For this search process, we give population  $P$  as input. For each particle  $i$ , in the population  $P$ , we select the random portion of *pbest* of size 0.25 times current particle *pbest*. Then we call

the flipping process for this random portion of  $pbest$  to add relevant non-selected features and remove the redundant features. If the new  $pbest$  is better than old  $pbest$ , then we update  $pbest$  to new  $pbest$  otherwise the old  $pbest$  is maintained.  $gbest$  is updated, if new  $gbest$  is better than old  $gbest$ , otherwise it not updated. Then the search process follows the Local Search 1, which we have discussed in JLOPSO 1 and the updated  $gbest$  is returned as output.



## CHAPTER 4

### IMPLEMENTATION AND RESULT ANALYSIS

#### 4.1 DATASET DESCRIPTION

A microarray database is a repository containing microarray gene expression data. Higher dimension of microarray data leads to more comprising time and it also contains noise and irrelevant genes which has negative effect in developing a drug or for disease recovery. Microarray data is used to evaluate in the proposed method. Table 4.1 shows the dataset implemented in this work.

**Table 4.1** Table showing the description about each data set

Datasets	# of features	#of Samples	# of Classes
LYMPHOMA	4026	66	3
LSVT	310	126	2
COLON	2000	62	2

#### 4.2 Metrics for Evaluation

- No. of features selected
- Accuracy

#### 4.3 JUMP LOCAL OPTIMA PARTICLE SWARM OPTIMIZATION I (JLPSO I)

This section pictorially shows the implementation and the results obtained from the Jump Local Optima Particle Swarm Optimization 1.

```

C=C+1

    GENE1835X  GENE1836X  GENE1865X  GENE1380X  ...  GENE2X  GENE48X  GENE47X  class
0      0.46      0.70      0.67      -0.23  ...   1.37     -0.04     0.16     1.0
1      0.02      0.59      0.45      0.55  ...  -0.05     -0.14    -1.15     1.0
2     -0.32     -0.63     -0.46     -0.28  ...   1.40      0.29     0.25     1.0
3     -0.51     -0.45     -0.16     -0.51  ...   1.24      0.05     0.70     1.0
4      0.20      0.13      0.20      0.09  ...  -0.72     -0.04    -0.22     1.0

[5 rows x 4027 columns]
No. of Features selected with threshold :4026.000000
No. of Sub-swarms 1
No. of particles 201
Population size in sub-swarms [201]
Initializing.....
Global best particle accuracy in sub-swarm 0 is 0
Initializing Sub-Swarm 0 Finished
Iteration # 1
Population size in sub-swarms [201]
Total No.of particles 201
Gbest 0.000000
0.9848484848484849 0 0
Global Best position in sub-swarm 0changed

1.0 0 0.9848484848484849
Global Best position in sub-swarm 0changed

0.9848484848484849 0 1.0

```

**Figure 4.1** JLOPSO I execution for Lymphoma Dataset

The figure 4.1 shows the no of features selected with threshold, number of Sub-Swarms, population size in Sub-Swarm and the global best position for Lymphoma dataset in the execution of JLOPSO I algorithm.

```

0.7588045101290323 0 0.8870967741935484
0.8225806451612904 0 0.8870967741935484
0.8870967741935484 0 0.8870967741935484
0.8064516129032258 0 0.8870967741935484
0.7741935483870968 0 0.8870967741935484
0.8709677419354839 0 0.8870967741935484
0.8225806451612904 0 0.8870967741935484
0.8548387096774194 0 0.8870967741935484
0.7903225806451613 0 0.8870967741935484
0.8387096774193549 0 0.8870967741935484
0.7903225806451613 0 0.8870967741935484
0.7741935483870968 0 0.8870967741935484
0.8387096774193549 0 0.8870967741935484
0.7903225806451613 0 0.8870967741935484
Relative Convergence:
[0.]
Relative Divergence:
[-0.32307653]
gbest: 0.8870967741935484
Total time taken: 0.270611
Save reduced dataset at iteration: 1
Total No. of Features: 2000
NO. of Features Selected: 62
Selected Features: [244.0, 896.0, 61.0, 1247.0, 1292.0, 624.0, 74.0, 1029.0, 1226.0, 1580.0, 124.0, 1038.0, 693.0, 1327.0, 1911.0, 1807.0, 697.0, 50.0, 494.0,
Iteration # 2
Population size in sub-swarms [100]
Total No.of particles 100
Gbest 0.887097
0.7741935483870968 0 0.8870967741935484
0.8225806451612904 0.7419354838709677 0.8870967741935484
0.7419354838709677 0.8064516129032258 0.8870967741935484
0.7903225806451613 0.8064516129032258 0.8870967741935484
0.7903225806451613 0.8225806451612904 0.8870967741935484

```

**Figure 4.2** JLOPSO I execution for Colon Dataset

The figure 4.2 depicts in execution of JLOPSO I, the total no of features of the dataset, the number of features selected and the global best for Colon dataset.

## 4.4 JUMP LOCAL OPTIMA PARTICLE SWARM OPTIMIZATION II (JLPSO II)

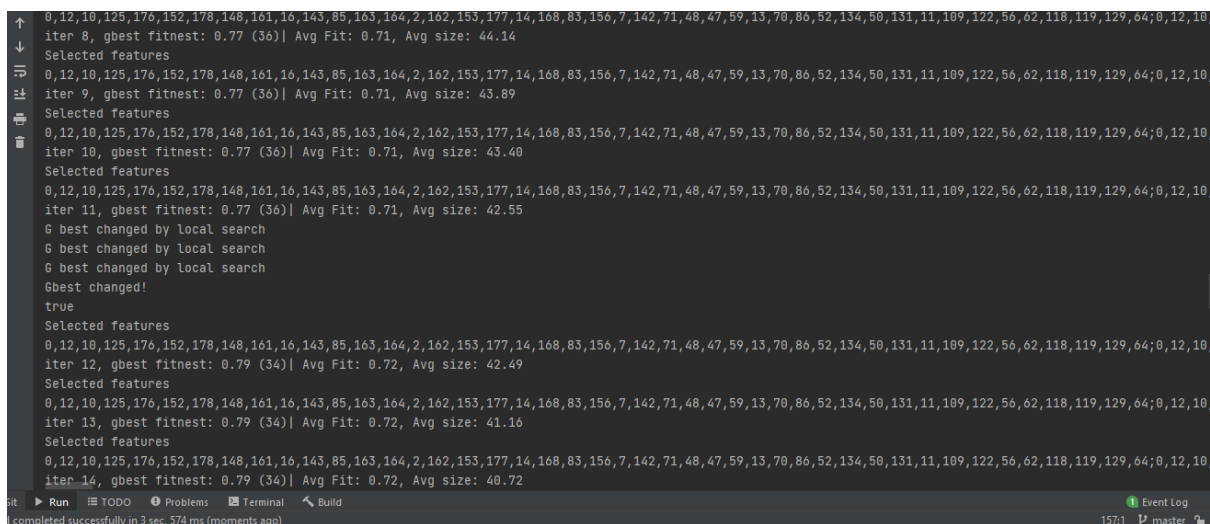
This section pictorially shows the implementation and the results obtained from the Jump Local Optima Particle Swarm Optimization 2.



```
Nbr of features: 4026
Nbr of instances: 66
Class: 0.0 nbr. of instances: 46 (69.70%)
Class: 1.0 nbr. of instances: 9 (13.64%)
Class: 2.0 nbr. of instances: 11 (16.67%)
Variable length PSO with size division 12
Max #Iterations: 30
Features in Descending order of SU
2775,733,2800,2735,3738,2721,2802,2746,2732,760,2801,2778,2761,3759,3737,3734,2792,2874,2803,2817,759,2737,2749,2738,757,2893,2758,2806,2770,2815,2779,2818,26
```

**Figure 4.3** JLPSO II execution for Lymphoma Dataset

The figure 4.3 shows the execution of JLPSO II for Lymphoma Dataset. Which shows the number of features of the dataset, number of instances in the dataset, and the descending order of the features in the dataset according to their values of Symmetrical Uncertainty.



```
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 8, gbest fitness: 0.77 (36)| Avg Fit: 0.71, Avg size: 44.14
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 9, gbest fitness: 0.77 (36)| Avg Fit: 0.71, Avg size: 43.89
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 10, gbest fitness: 0.77 (36)| Avg Fit: 0.71, Avg size: 43.40
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 11, gbest fitness: 0.77 (36)| Avg Fit: 0.71, Avg size: 42.55
G best changed by local search
G best changed by local search
G best changed by local search
Gbest changed!
true
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 12, gbest fitness: 0.79 (34)| Avg Fit: 0.72, Avg size: 42.49
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 13, gbest fitness: 0.79 (34)| Avg Fit: 0.72, Avg size: 41.16
Selected features
0,12,10,125,176,152,178,148,161,16,143,85,163,164,2,162,153,177,14,168,83,156,7,142,71,48,47,59,13,70,86,52,134,50,131,11,109,122,56,62,118,119,129,64;0,12,10
iter 14, gbest fitness: 0.79 (34)| Avg Fit: 0.72, Avg size: 40.72
```

**Figure 4.4** JLPSO II execution for LSVT Dataset

The figure 4.4 shows the execution of JLPSO II for LSVT Dataset which shows the iteration count, fitness of global best, selected features, and whether the global best changed by local search or not.

## 4.5 RESULTS

**Table 4.2** Table showing the number features selected for each dataset by JLOPSO I

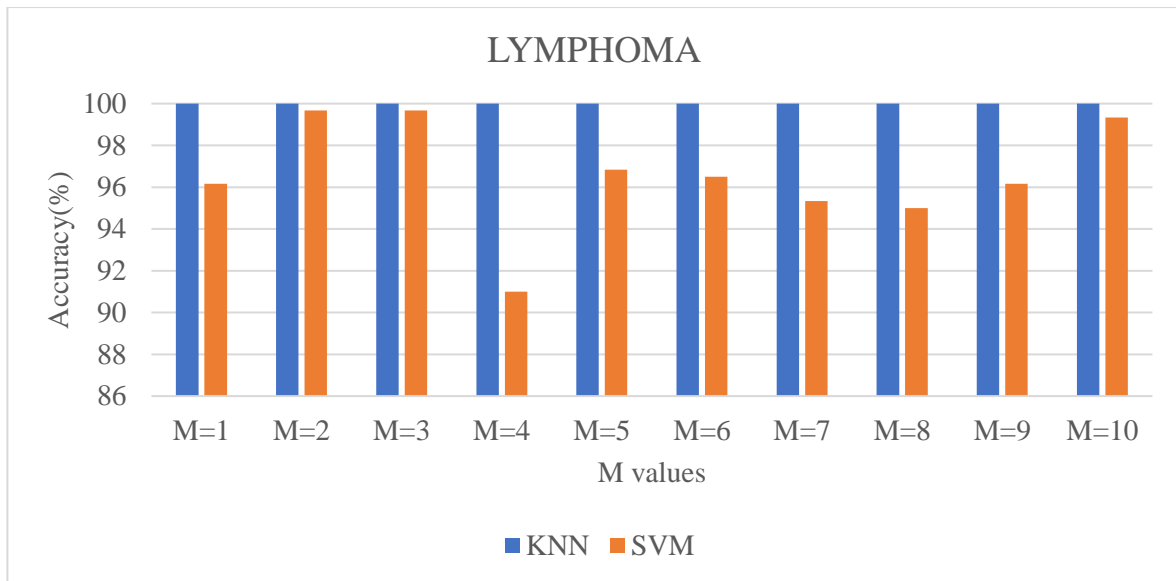
Dataset	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
<b>LYM PHOMA</b>	30.97 (100)	28.27 (100)	25.17 (100)	13.40 (100)	19.63 (100)	12.03 (100)	8.07 (100)	16.20 (100)	13.77 (100)	9.63 (100)
<b>LSVT</b>	3.83 (78.36)	6.97 (80.63)	4.80 (82.48)	8.47 (84.07)	5.50 (79.95)	6.57 (83.33)	5.37 (79.81)	6.03 (82.65)	9.97 (85.74)	9.23 (87.81)
<b>COLON</b>	32.90 (95.27)	34.57 (98.76)	20.57 (94.89)	23.70 (97.53)	28.50 (99.46)	33.67 (97.69)	18.00 (96.13)	30.43 (98.12)	15.87 (98.82)	12.07 (97.37)

The table 4.2 shows the average number of features selected by executing the JLOPSO I algorithm for Lymphoma, LSVT and Colon datasets. From this table we conclude that we have got least number of features selected.

**Table 4.3** Table showing the number features selected for each dataset by different algorithms

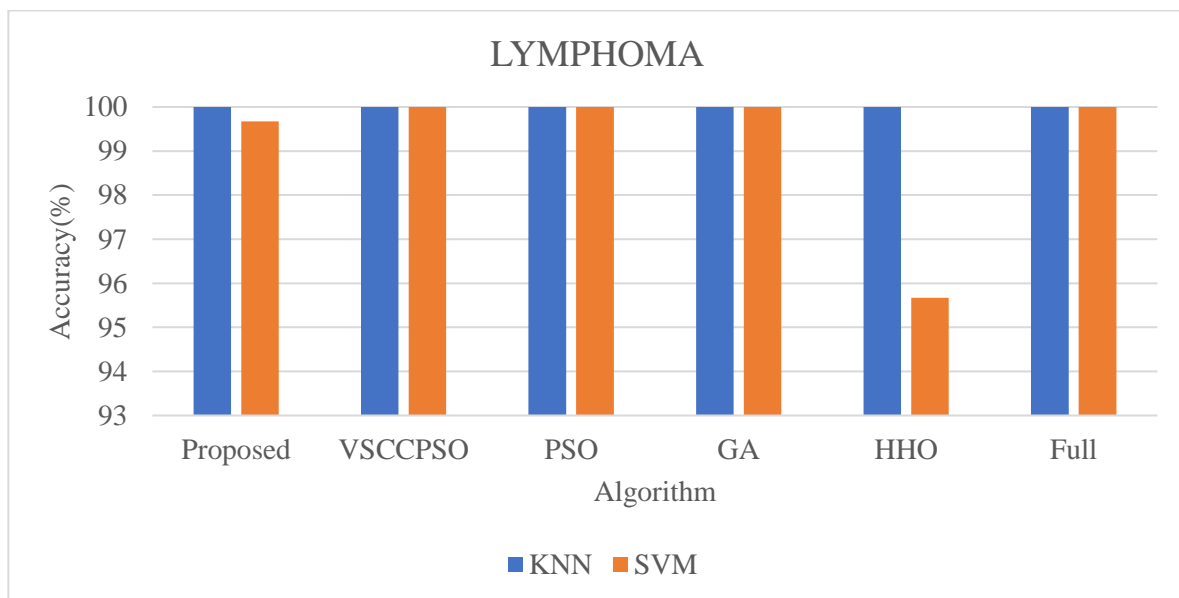
Dataset	Classifier/ Algorithm	PROPOSED	VSCCPSO	PSO	GA	HHO	FULL
<b>LYMPHOMA</b>	<b>KNN</b>	13.77 (100)	56 (100)	1791.67 (100)	1778.8 (100)	105.23 (100)	4026 (100)
	<b>SVM</b>	25.17 (99.67)	56 (100)	1791.67 (100)	1778.8 (100)	105.23 (95.67)	4026 (100)
<b>LSVT</b>	<b>KNN</b>	9.23 (87.81)	30.43 (87.2)	117.53 (59.52)	130.6 (71.77)	16.7 (70.42)	310 (51.59)
	<b>SVM</b>	6.57 (79.3)	22.5 (75.88)	117.53 (69.3)	130.6 (74.39)	16.7 (67.37)	310 (81.58)
<b>COLON</b>	<b>KNN</b>	28.5 (99.46)	35.93 (98.17)	927.47 (78.44)	941.6 (80.38)	82.13 (85.7)	2000 (69.35)
	<b>SVM</b>	20.57 (89.65)	24.93 (86.31)	927.47 (84.21)	941.6 (82.28)	82.13 (82.63)	2000 (84.21)

The table 4.3 shows the number of features selected by executing the algorithms PROPOSED, VSCCPSO, PSO, GA, and HHO for datasets Lymphoma, LSVT and Colon. On comparing with other algorithms our proposed algorithm selected least number of features, this shows the correctness of the algorithm.



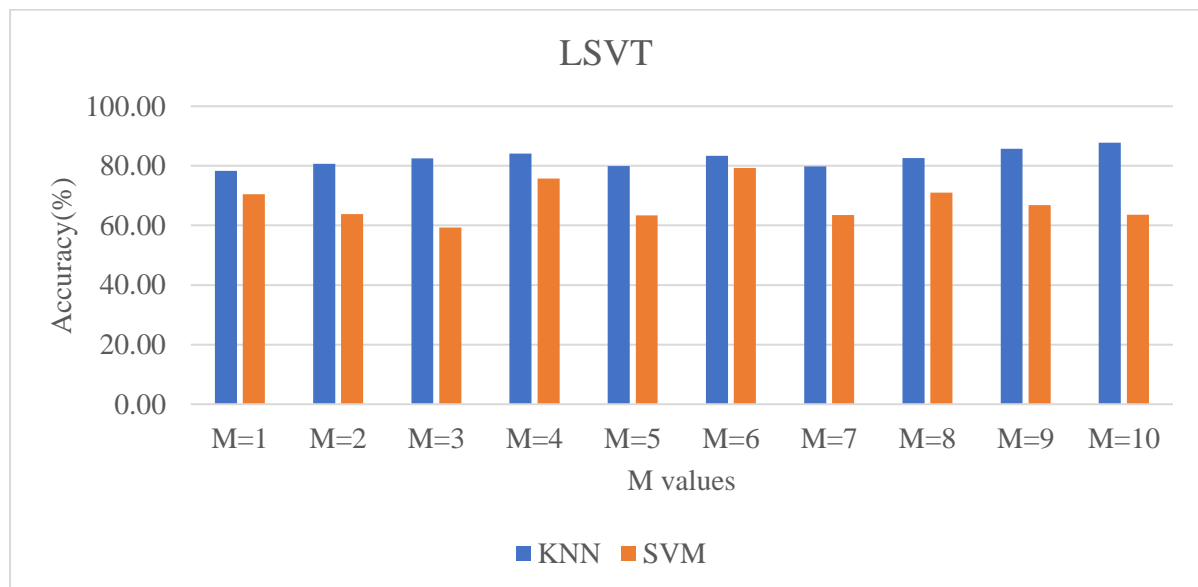
**Figure 4.5** Accuracy Graph for Different M Values for Lymphoma dataset

The figure 4.5 shows about the accuracy for different M values for Lymphoma dataset on executing the JLOPSO I algorithm. Here we have got full accuracy for the KNN classifier for each M value.



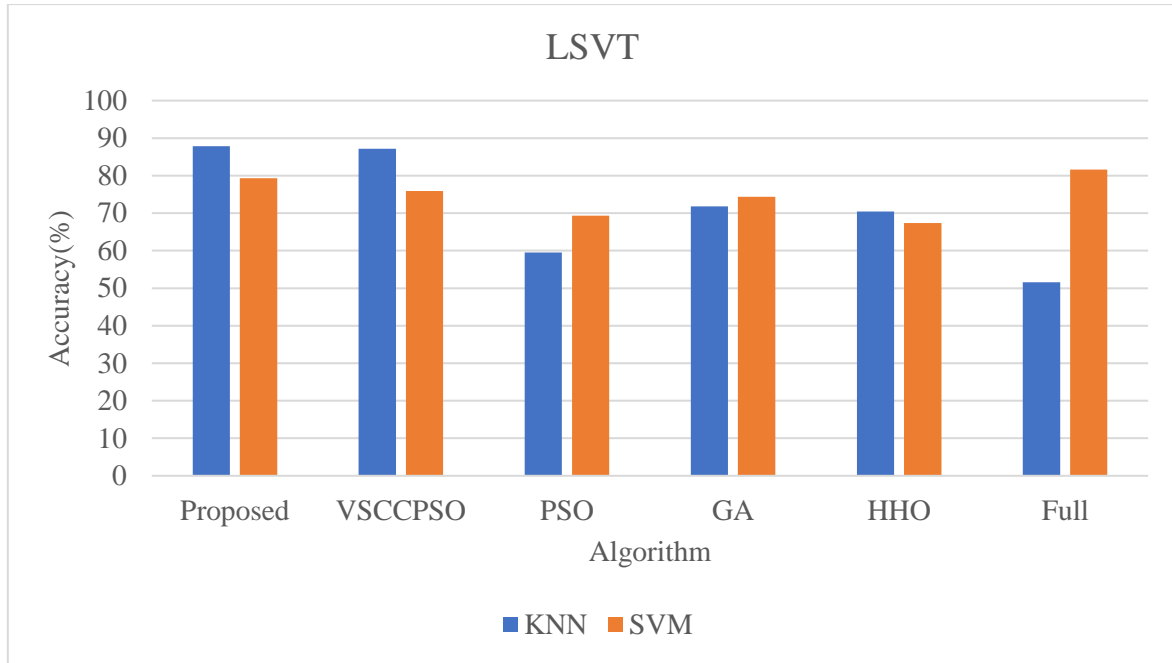
**Figure 4.6** Comparing the accuracies for Lymphoma Dataset

The figure 4.6 shows the classification accuracy comparisons of the feature sets from each of the methods from Lymphoma dataset on executing the JLOPSO I algorithm and other existing algorithms.



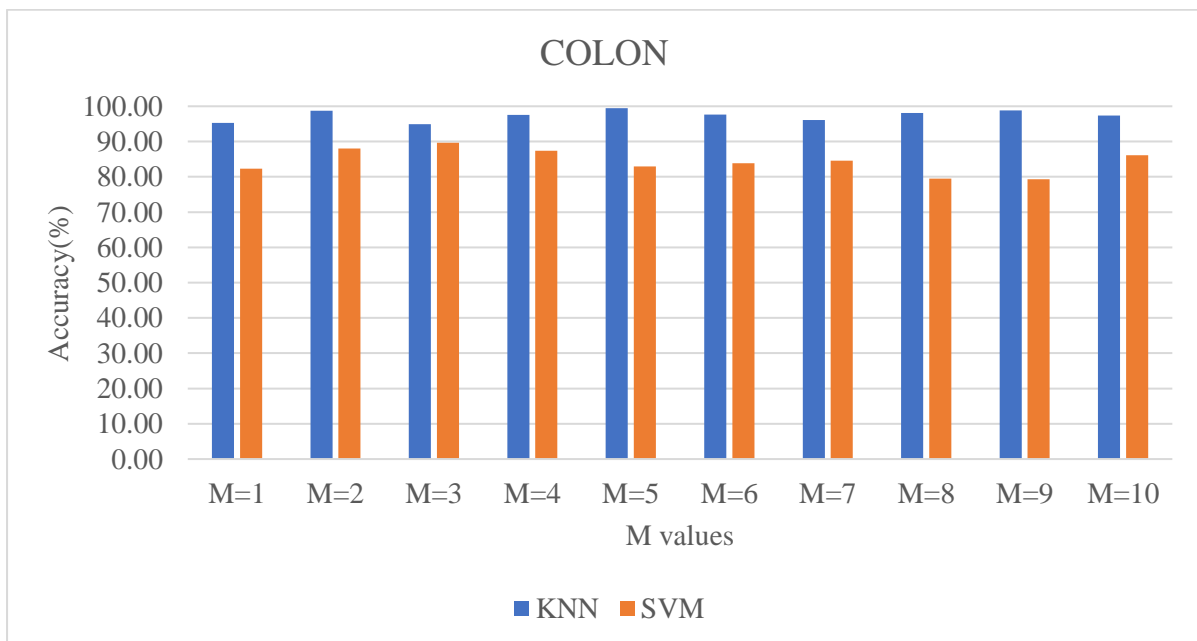
**Figure 4.7** Accuracy Graph for Different M Values for LSVT dataset

The figure 4.7 shows about the accuracy for different M values for LSVT dataset on executing the JLOPSO I algorithm. Here we have got higher accuracy in all cases.



**Figure 4.8** Comparing the accuracies for LSVT Dataset

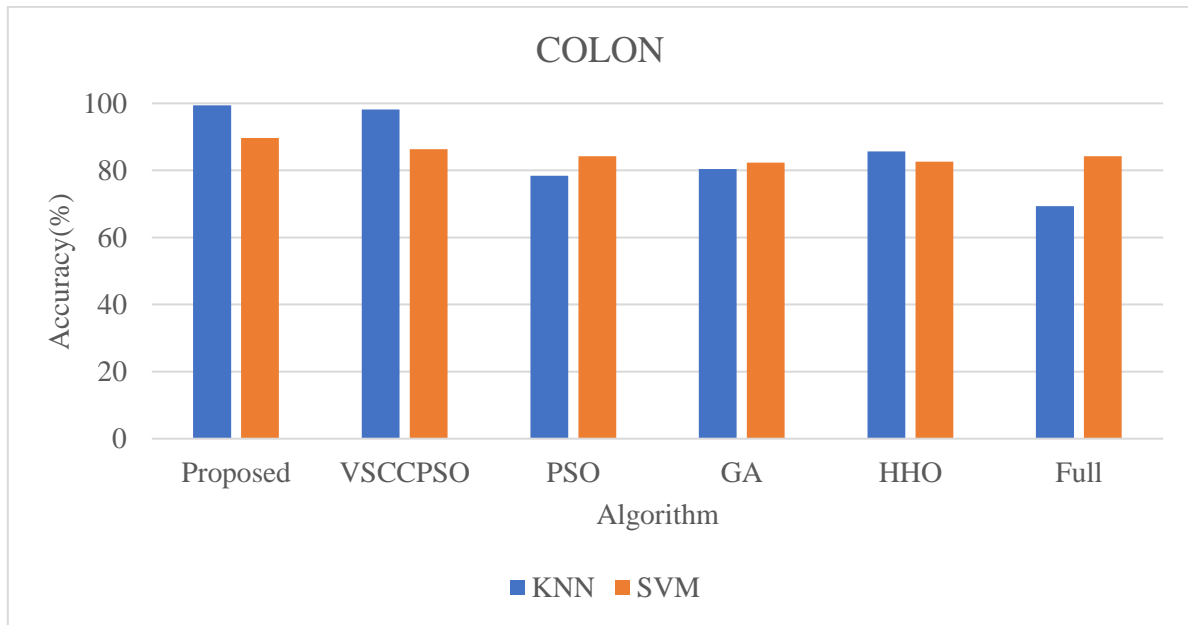
The figure 4.8 shows the classification accuracy comparisons of the feature sets from each of the methods from LSVT dataset on executing the JLOPSO I algorithm and other existing algorithms. And we have got higher accuracy for JLOPSO I compared to other algorithms.



**Figure 4.9** Accuracy Graph for Different M Values for COLON dataset



The figure 4.9 shows about the accuracy for different M values for LSVT dataset on executing the JLOPSO I algorithm.



**Figure 4.10** Comparing the accuracies for Colon Dataset

The figure 4.10 shows the classification accuracy comparisons of the feature sets from each of the methods from LSVT dataset on executing the JLOPSO I algorithm and other algorithms VSCCPSO, PSO, GA, and HHO. The existing algorithms got less accuracy compared with our proposed algorithm.

**Table 4.4** Table showing the number features selected for each dataset by

JLOPSO II

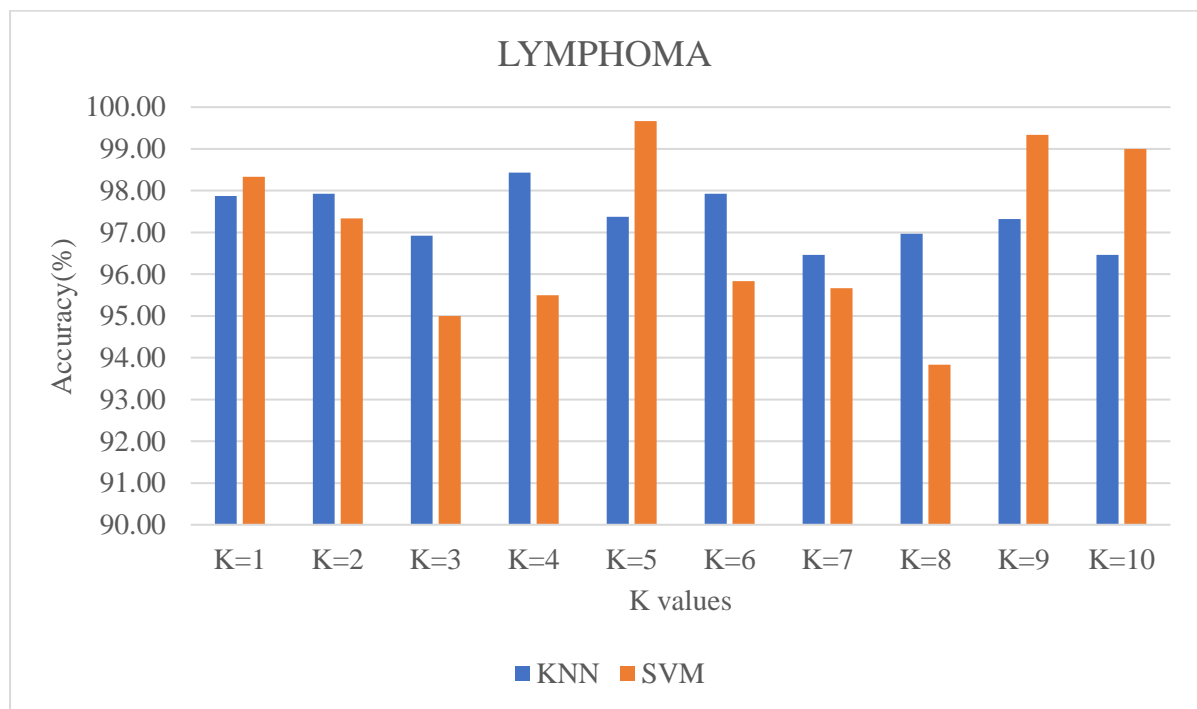
Dataset	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9	K=10
<b>LYM PHOMA</b>	11.67 (97.88)	20.80 (97.93)	20.00 (96.92)	6.67 (98.43)	8.83 (97.37)	13.13 (97.93)	5.03 (96.46)	8.17 (96.97)	14.53 (97.32)	10.93 (96.46)
<b>LSVT</b>	33.00 (53.97)	37.60 (65.18)	28.03 (70.32)	31.50 (61.72)	39.87 (58.97)	38.57 (68.25)	35.43 (66.38)	37.63 (65.13)	33.47 (66.67)	37.00 (66.67)
<b>COLON</b>	33.63 (84.73)	28.93 (81.46)	27.33 (82.37)	36.90 (82.85)	36.33 (88.39)	34.67 (88.10)	29.83 (84.19)	30.27 (78.71)	36.03 (80.86)	43.30 (82.21)

The table 4.3 shows number of features selected by executing the JLOPSO II algorithm for Lymphoma, LSVT and Colon datasets.

**Table 4.5** Table showing the number features selected for each dataset by different algorithms

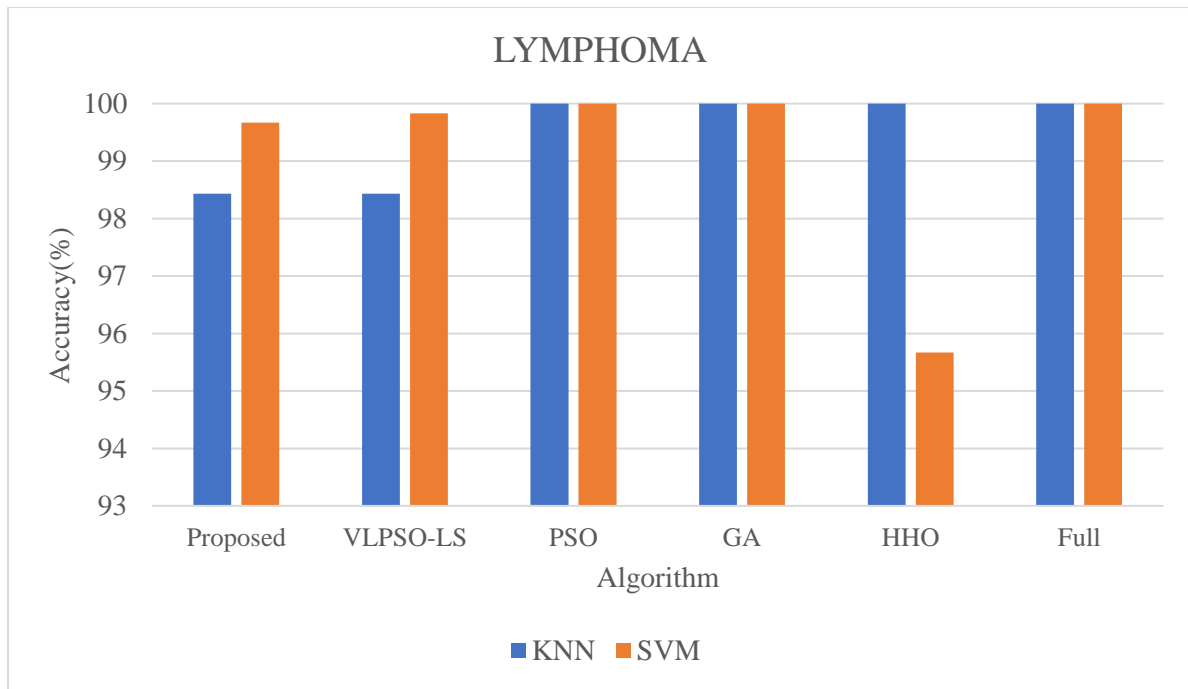
Dataset	Classifier/ Algorithm	PROPOSED	VLPSO- LS	PSO	GA	HHO	FULL
<b>LYMPHO MA</b>	<b>KNN</b>	6.67 (98.43)	32.8 (98.43)	1791.67 (100)	1778.8 (100)	105.23 (100)	4026 (100)
	<b>SVM</b>	8.83 (99.67)	32.8 (99.83)	1791.67 (100)	1778.8 (100)	105.23 (95.67)	4026 (100)
<b>LSVT</b>	<b>KNN</b>	28.03 (70.32)	27.2 (70.37)	117.53 (59.52)	130.6 (71.77)	16.7 (70.42)	310 (51.59)
	<b>SVM</b>	37.63 (81.76)	37.63 (81.76)	117.53 (69.3)	130.6 (74.39)	16.7 (67.37)	310 (81.58)
<b>COLON</b>	<b>KNN</b>	36.33 (88.39)	37 (88.71)	927.47 (78.44)	941.6 (80.38)	82.13 (85.7)	2000 (69.35)
	<b>SVM</b>	36.33 (88.94)	37 (89.47)	927.47 (84.21)	941.6 (82.28)	82.13 (82.63)	2000 (84.21)

The table 4.5 shows that the number of features selected by executing the JLOPSO II and different existing algorithms. When compare among these algorithms, the proposed algorithm have selected least number of features which depicts the good working of the algorithm.



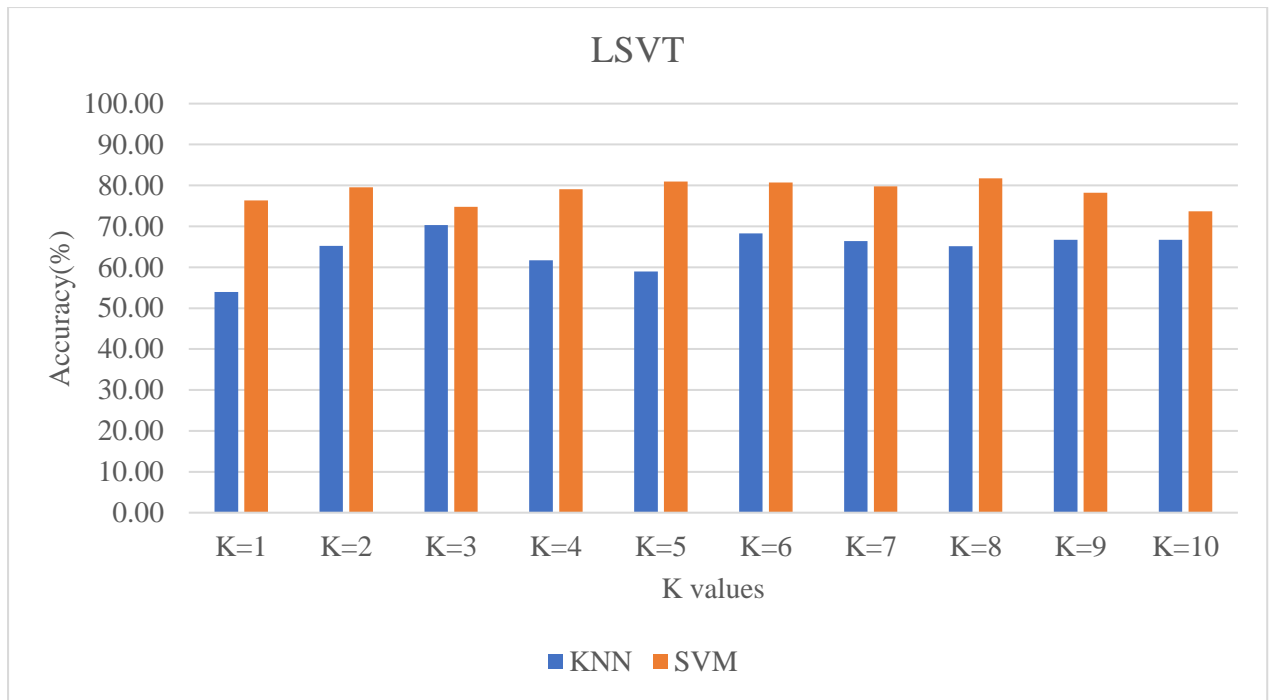
**Figure 4.11** Accuracy Graph for Different K Values for LYMPHOMA dataset

The figure 4.11 shows about the accuracy for fiffereent K values for Lymphoma dataset on executing the JLOPSO II algorithm.



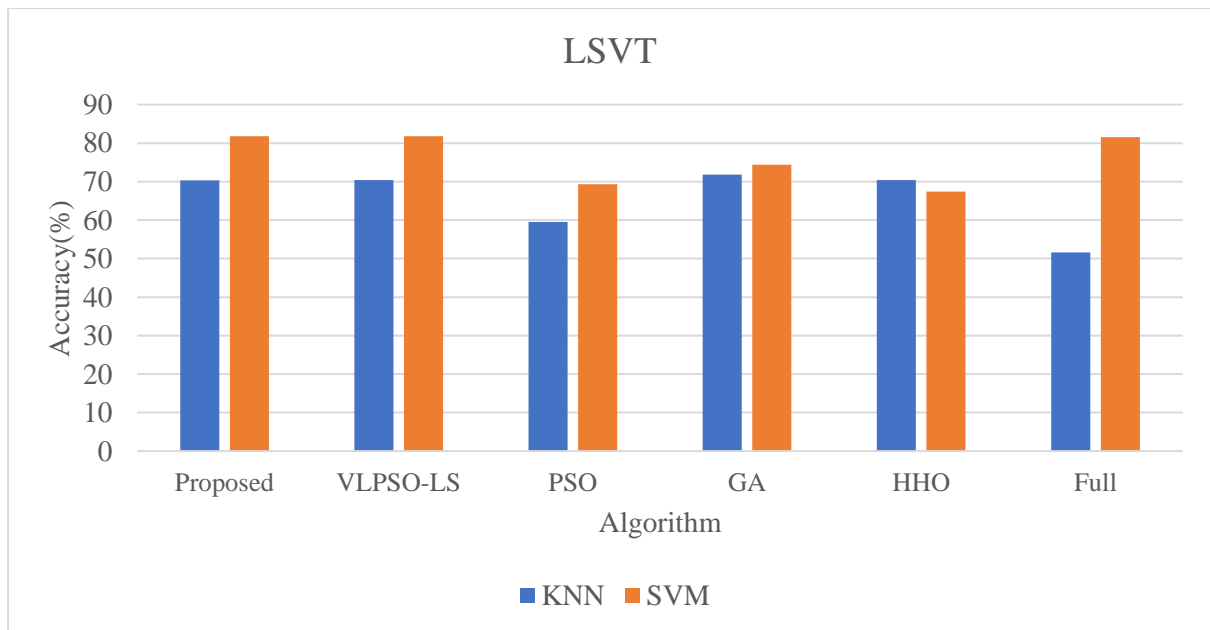
**Figure 4.12** Comparing the accuracies for Lymphoma Dataset

The figure 4.12 shows the classification accuracy comparisons of the feature sets from each of the methods from Lymphoma dataset on executing the JLOPSO II algorithm and other algorithms.



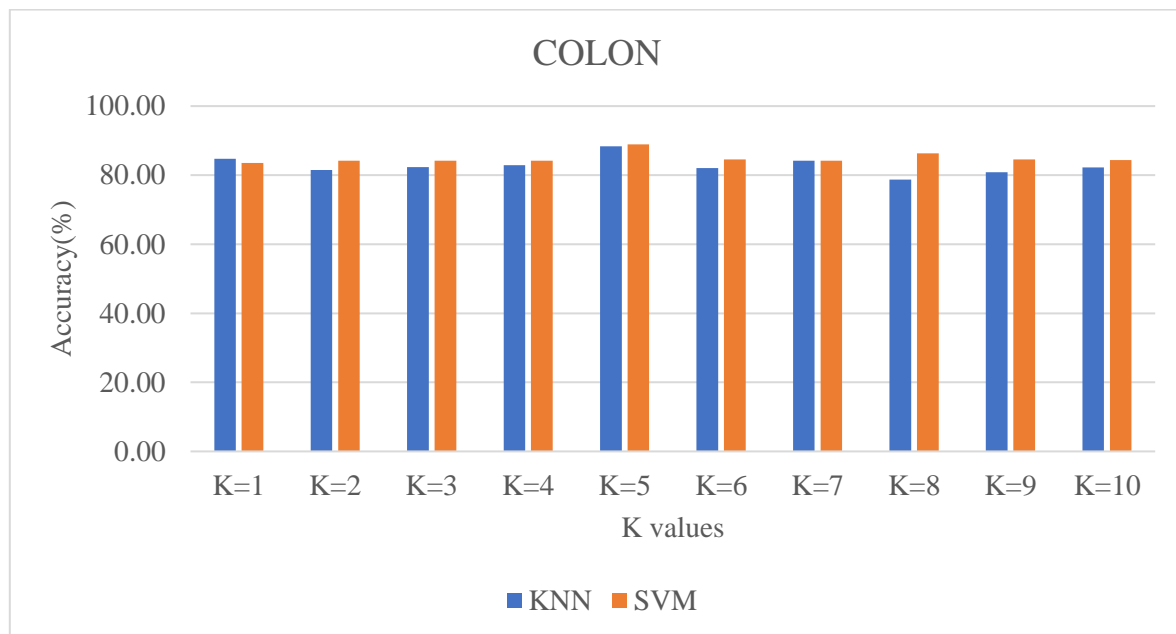
**Figure 4.13** Accuracy Graph for Different K Values for LSVT dataset

The figure 4.13 shows the accuracy for different K values for LSVT dataset on executing the JLOPSO II algorithm.



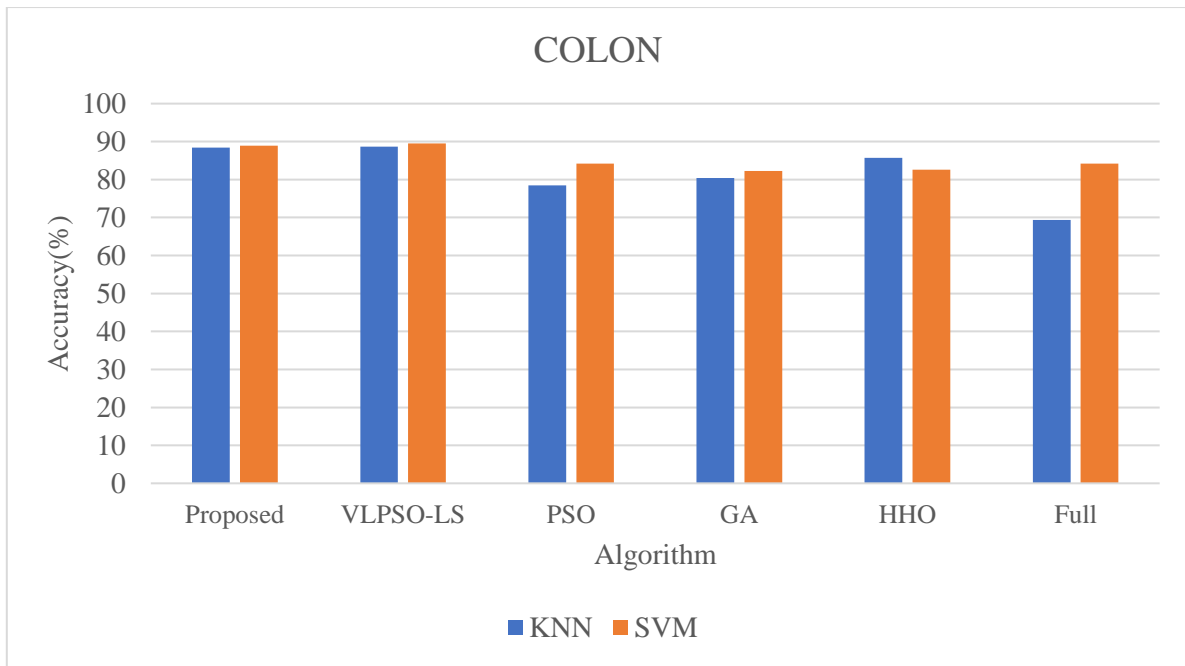
**Figure 4.14** Comparing the accuracies for LSVT Dataset

The figure 4.14 shows the classification accuracy comparisons of the feature sets from each of the methods from Lymphoma dataset on executing the JLOPSO II algorithm and other existing algorithms.



**Figure 4.15** Accuracy Graph for Different K Values for COLON dataset

The figure 4.15 shows the accuracy for different K values for Colon dataset on executing the JLOPSO II algorithm.



**Figure 4.16** Comparing the accuracies for Colon Dataset

The figure 4.16 shows the classification accuracy comparisons of the feature sets from each of the methods from Lymphoma dataset on executing the JLOPSO II algorithm and VLPSO-LS, PSO, GA and HHO. The JLOPSO II have got higher accuracy compared to other algorithms.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE WORK**

#### **5.1 CONCLUSION**

It is clearly seen that the accuracy for reduced datasets are certainly higher than that of the entire datasets. This ensures the correctness of the proposed algorithm. Thus the difficult process of feature selection is done in a simple way in two different approaches and the importance of Particle Swarm Optimization is clearly explained. The essential parameters that improve the Particle Swarm Optimization are clearly shown and their importance is neatly established.

#### **5.2 FUTURE WORK**

The reduced dataset from the two different approaches can be ensembled into one for classification problems or any other machine learning related works. This can improve the efficiency of the feature set in various processes since the ensembled set will have higher accuracy than that of the individual sets.



## REFERENCES

1. Xian-fang Song, Yong Zhang, Member, IEEE, Yi-nan Guo, Xiao-yan Sun, Yong-li Wang, “Variable-size Cooperative Coevolutionary Particle Swarm Optimization for Feature Selection on High-dimensional Data”, IEEE Transactions On Evolutionary Computation, Vol. 24, No. 5, pp. 882-895, October 2020.
2. Binh Tran , Member, IEEE, Bing Xue, Member, IEEE, and Mengjie Zhang , Senior Member, IEEE, “Variable-Length Particle Swarm Optimization for Feature Selection on High-Dimensional Classification”, IEEE Transactions On Evolutionary Computation, Vol. 23, No. 3, Pp. 473-487,june 2019
3. Simon Fong, Raymond Wong, and Athanasios V. Vasilakos, “Accelerated PSO Swarm Search Feature Selection for Data Stream Mining Big Data”, IEEE Transaction on Services Computing, vol. 9, no. 1, January/February 2016, pp. 33-45.
4. Yong Zhang, Dun-wei, and Jian Cheng, “Multi-Objective Particle Swarm Optimization Approach for Cost-Based Feature Selection in Classification”, IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 14, no. 1, January/February 2017, pp. 64-75.
5. Binh Tran, Bing Xue, and Mengjie Zhang, “A New Representation in PSO for Discretization-Based Feature Selection”, IEEE Transactions on Cybernetics, vol. 48, no. 6, June 2018, pp. 1733-1746.
6. Luca Bravi, Veronica Piccialli, and Marco Sciandrone, “An Optimization-Based Method for Feature Ranking in Nonlinear Regression Problems”, IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 4, April 2017, pp. 1005-1010.
7. Zhi-Zhong Liu, Yong Wang, Shengxiang Yang and Ke Tang, “An Adaptive Framework to Tune the Coordinate Systems in Nature-Inspired

- Optimization Algorithms”, IEEE Transactions on Cybernetics, vol. 49, no. 4, April 2019, pp. 1403- 1416.
8. Kamlesh Mistry, Li Zhang, Siew Chin Neoh, Chee Peng Lim, and Ben Fielding, “A Micro-GA Embedded PSO Feature Selection Approach to Intelligent Facial Emotion Recognition”, IEEE Transactions on Cybernetics, vol. 47, no. 6, June 2017, pp. 1496-1509.
  9. Makoto Yamada, Jiliang Tang, Jose Lugo-Martinez, Ermin Hodzic, Raunak Shrestha, Avishek Saha, Hua Ouyang, Dawei Yin, Hiroshi Mamitsuka, Cenk Sahinalp, Predrag Radivojac, Filippo Menczer, and Yi Chang, “Ultra High-Dimensional Nonlinear Feature Selection for Big Biological Data”, IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 7, July 2018, pp. 1352-1365.
  10. Narges Armanfard, James P. Reilly, and Majid Komeili, “Local Feature Selection for Data Classification”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 6, June 2016, pp. 1217-1227.
  11. Bin Hu, Yongqiang Dai, Yun Su, Philip Moore, Xiaowei Zhang, Chengsheng Mao, Jing Chen, and Lixin Xu, “Feature Selection for Optimized High-Dimensional Biomedical Data Using an Improved Shuffled Frog Leaping Algorithm”, IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 15, no. 6, November/December 2018, pp. 1765-1773.
  12. Jorge González-López, Sebastián Ventura, and Alberto Cano, “Distributed Selection of Continuous Features in Multilabel Classification Using Mutual Information”, IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 7, July 2020, pp. 2280-2293.
  13. Mohammed A. Ambusaidi, Xiangjian He, Priyadarsi Nanda, and Zhiyuan Tan, “Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm”, IEEE Transactions on Computers, vol. 65, no. 10, October 2016, pp. 2986-2998.

14. Lefei Zhang, Qian Zhang, Bo Du, Xin Huang, Yuan Yan Tang and Dacheng Tao, "Simultaneous Spectral-Spatial Feature Selection and Extraction for Hyperspectral Images", *IEEE Transactions on Cybernetics*, vol. 48, no. 1, January 2018, pp. 16-28.
15. L. Yu and H. Liu, "Efficient Feature Selection via Analysis of Relevance and Redundancy," *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, 2004.
16. S. Alelyani, J. Tang, and H. Liu, "Feature Selection for Clustering: A Review," in: C. Aggarwal and C. Reddy (eds.), *Data Clustering: Algorithms and Applications*, CRC Press, 2013.
17. H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, 2005.
18. N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "MIFS-ND: A mutual information-based feature selection method", *Expert Systems with Applications*, vol. 41, issue 14, pp. 6371–6385, 2014.
19. A. S. U. Kamath, K. De Jong, and A. Shehu, "Effective automated feature construction and selection for classification of biological sequences," *PLoS One*, vol. 9, no. 7, 2014, Art. ID e99982.
20. W. A. Albukhanajer, J. A. Briffa, and Y. Jin, "Evolutionary multiobjective image feature extraction in the presence of noise," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1757–1768, Sep. 2015.
21. M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, nos. 1–4, pp. 131–156, 1997.
22. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
23. Rami Sihwail, Khairuddin Omar, Khairul Akram Zainol Ariffin, And Mohammad Tubishat, "Improved Harris Hawks Optimization Using Elite Opposition-Based Learning and Novel Search Mechanism for Feature Selection", *IEEE Access*, vol. 8, pp. 121127– 121145, July. 2020.