Palestine Polytechnic University

Deanship of Graduates Studies and Scientific Research

Master of Informatics

Parallel Scatter Search Algorithm for Feature Selection in High Dimensional Datasets

Submitted by: Safa Ibrahim Adi

Supervisor: Dr. Mohammed Aldasht

In partial Fulfillment of the Requirements for the Master Degree of Science in Informatics

June, 2017

The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled: **Parallel Scatter Search Algorithm for Feature Selection in High Dimensional Datasets**, submitted by **Safa Ibrahim Adi** in partial fulfillment of the requirements for the Master degree of Science in Informatics.

Graduate advisory committee:

Dr. Mohammed Aldasht (Supervisor), Palestine Polytechnic University.

Signed: Date:

Dr. Mahmoud Alsaheb (Internal committee member), Palestine Polytechnic University.

Signed: Date:

Prof. Dr. Mohammed Awad (External committee member), Arab American University

Signed: Date:

Thesis Approved

Dr. Murad Abusubaih

Dean of Graduate Studies and Scientific Research

Palestine Polytechnic University

Signed: Date:

Declaration

I declare that this thesis titled, **'Parallel Scatter Search Algorithm for Feature Selection in High Dimensional Datasets'** is my own original work, and all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Safa Ibrahim Adi

Signed

Date

Statement of permission to use

In presenting this thesis in partial fulfillment of the requirement for the master degree in Informatics at Palestine Polytechnic University. I agree that the library shall make it available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by main supervisor, or in in his absence by the Dean of Graduate Studies and Scientific Research, when in the opinion of either the proposed use of the material is for scholarly purpose.

Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Safa Ibrahim Adi

Signed

Date

Dedication

I dedicate this work to my family.

Acknowledgement

I would like to express my sincere gratitude to my supervisor Dr. Mohammed Aldasht. He continuously provided me with help, encouragement and knowledge.

I would like to thank the examination committee, Dr. Mohammed Awad and Dr. Mahmoud Alsaheb for their insightful comments.

I thank my colleagues Dr. Mohammed Abutaha and Anas Amro, they gave me a lot of knowledge and answered all my questions.

Abstract

Feature selection is one of a key success factor for classification problem in high dimensional datasets. This process aims to select the discriminative subsets of features in order to enhance the classification performance and reduce learning time.

In this thesis we introduce an approach for handling the classification problem in high dimensional datasets using scatter search algorithm with wrapper model. During our study we have implemented the sequential and the parallel versions of the scatter search algorithm.

The classification performance for the two versions are similar for Mushroom, Madelon, Gisette and Spambase datasets. For Arcene dataset the parallel version of the scatter search enhances the classification performance from **0.93** to **0.94** comparing with the sequential version of the scatter search.

Five benchmark datasets are used to evaluate our approach, all of them are two-class classification problem. They are: Mushroom, Spambase, Arcene, Madelon and Gisette. Three of them (Arcene, Madelon and Gisette) are feature selection challenge.

The obtained results indicate that the proposed approach is very efficient for feature selection process in high dimensional datasets; since the scatter search algorithm reduces the execution time and enhances the classification performance.

A comparative study is conducted with other research in the literature that uses the evolutionary algorithms to handle the classification problem in high dimensional datasets. Our proposed method is very efficient, it reduced the execution time for all the datasets that we use in our experiments, and enhanced the classification results, the classification results are ranged from **0.92** to **1.0** for the five datasets.

الملخص

تهدف هذه الدراسة إلى تحسن أداء عملية التصنيف عن طريق اختيار الصفات باستخدام خوارزمية البحث المبعثر، خاصة في حالة وجود عدد كبير جدا من الصفات والعينات في مجموعة البيانات. حيث أن اختيار الصفات يعتبر عامل النجاح الرئيسي في حل مشكلة التصنيف في مجموعات البيانات كثيرة الأبعاد. وتهدف هذه العملية إلى اختيار المجموعات الجزئية التمييزية من الصفات من أجل تعزيز أداء التصنيف وتقليل وقت تعلم خوارزمية التصنيف.

تشير النتائج التي تم تنفيذها على خمس مجموعات اختبارية مختلفة إلى أن النهج المقترح كان فعالا جدا في اختيار العناصر التمييزية من مجموعات البيانات ذات الأبعاد الكثيرة، وذلك بفضل خوارزمية البحث المبعثر التي تمكنت من تقليل وقت التنفيذ ورفع جودة التصنيف مقارنة مع الطرق المستخدمة في أبحاث أخرى والتي تعتمد على خوارزميات تطورية لمعالجة مشكلة التصنيف في مجموعات البيانات كثيرة الأبعاد.

خلال عملنا في هذه الرسالة قمنا ببرمجة نسختين من خوارزمية البحث المبعثر، النسخة الأولى للتنفيذ التتابعي أو (Sequential) والنسخة الأخرى للتنفيذ على التوازي أو (Parallel). النسخة المتوازية من خوارزمية البحث المبعثر لديها نفس النتائج تقريبا لأداء التصنيف مثل النسخة المتسلسلة من نفس الخوارزمية. ولكن من ناحية أخرى، خفضت خوارزمية البحث المبعثر المتوازية وقت التنفيذ لجميع مجموعات البيانات التي نستخدمها في تجاربنا بشكل ملحوظ.

Contents

List of Figures

List of Tables

Abbreviations

ACC	**A**ccuracy
ACO	**A**nt **C**olony **O**ptimization
AI	**A**rtificial **I**ntelligence
AUC	**A**rea **U**nder **C**urve
CCA	**C**anonical **C**orrelation **A**nalysis
EA	**E**volutionary **A**lgorithm
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GA	**G**enetic **A**lgorithm
GC	**G**reedy **C**ombination
GPSO	**G**eometric **P**article **S**warm **O**ptimization
HUX	**H**alf **U**niform **C**rossover
KKT	**K**arush **K**uhn **T**ucker
LDA	**L**inear **D**iscriminative **A**nalysis
LOOCV	**L**eave **O**ne **O**ut **C**ross **V**alidation
ML	**M**achine **L**earning
MPI	**M**essage **P**assing **I**nterface
NP	**N**ondeterministic **P**olynomial time
PCA	**P**rinciple **C**omponent **A**nalysis
PET	**P**arallel **E**xecution **T**ime
PSO	**P**article **S**warm **O**ptimization
PVM	**P**arallel **V**irtual **M**achine
RF	**R**andom **F**orest
RGC	**R**educed **G**reedy **C**ombination
SET	**S**equential **E**xecution **T**ime
SS	**S**catter **S**earch
SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive

Chapter 1

Introduction

Nowadays, many disciplines have to deal with high dimensional datasets which involve a large number of features. Thus we need data preprocessing methods and data reduction models in order to simplify input data [1].

The selection of optimal feature subset is an optimization problem that proved to be NP-hard, complex, and time-consuming problem [2, 3]. As seen in Figure 1.1, there are two major approaches used to tackle the NP-hard problems: exact methods and metaheuristics. The former is exact methods allow exact solution to be found, but this approach is impractical since it is extremely time consuming for real world problems. The later is metaheuristics is used for solving complex and real world problems, because metaheuristics provide suboptimal (sometimes optimal) solution in reasonable time [2, 4, 5].

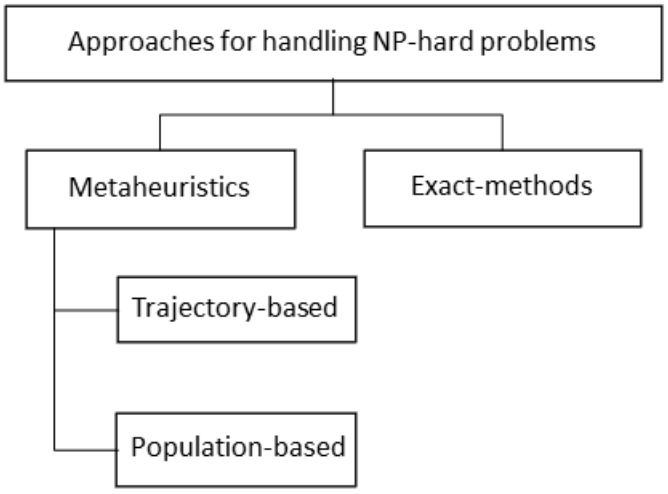

FIGURE 1.1: Approaches for handling NP-hard problems

Metaheuristics algorithms have proved to be suitable tools for solving the feature selection accurately and efficiently for high dimensions in large datasets [4, 6, 7].

The main problems that appear when dealing with high dimensional datasets are: 1) execution time because the complexity of the metaheuristics methods for feature selection is at least $O(n^2D)$ [8], where n is the number of instances and D is the number of features. 2) memory consumption since most methods for feature selection need to store the whole dataset in memory. Therefore, the researchers try to parallelize the sequential metaheuristics to improve their efficiency for feature selection in high dimensional datasets. They used many programming models and paradigms, such as MapReduce (Hadoop, spark), MPI, OpenMP, CUDA [2, 9, 10] for that purpose. Parallel computing can be process interaction (shared memory, message passing) or problem decomposition (task or data parallelization) [10].

Parallel implementations of metaheuristics are an effective alternative to speed up sequential metaheuristics; by reducing the search time of solutions for optimization problems. Furthermore, they lead to the more precise random algorithm and improve the quality of solutions [5].

Feature selection is a problem with high dimensional datasets. In order to make classification faster and more accurate, we need to select the subset of features that are discriminative. Evolutionary algorithms like *Genetic Algorithms* (GA), *Scatter Search* (SS) algorithm, *Particle Swarm Optimization* (PSO), *Ant Colony Optimization* (ACO), etc. are methods can be effective for this problem, but they require long execution time, also a large memory space [11]. To overcome these weaknesses, and to enhance the classification performance in high dimensional datasets, *Parallel Evolutionary Algorithms* (PEAs) can be used.

1.1 Thesis objective

The aim of this thesis, is to enhance the classification performance using feature selection method built on parallel scatter search algorithm, that uses wrapper approach for feature selection. Furthermore, this thesis aims to reduce the execution time comparing with the time needed by any other evolutionary algorithm.

1.2 Thesis hypothesis

In this thesis, we assume that using our parallel approach for feature selection in high dimensional datasets will lead to more possible solutions in less time that is needed

in sequential evolutionary algorithms. By using *Parallel Scatter Search algorithm*, our approach will reduce the execution time and enhance the classification performance for high dimensional datasets, comparing with other evolutionary algorithms (sequential and parallel).

1.3 Contributions

In this thesis, we will explore a new possibilities using the parallel evolutionary algorithms to assess the classification problem with the high dimensional datasets, more precisely we will:

- Build the model for feature selection using scatter search algorithm.
- Develop our parallel scatter search algorithm using master-worker model, to better explore the search space to find better solutions for the feature selection problem.
- Conduct comparisons with previous works using well known benchmarks to prove the effectiveness of our methodology.

1.4 Thesis organization

Our thesis is organized as follow, in Chapter 2, we will give a background, and review the main theoretical concepts are needed to understand the terminology of this thesis. In Chapter 3, we will state the previous works and literature review about parallel evolutionary algorithms for feature selection in high dimensional datasets. In Chapter 4, we will introduce our methodology that we use to enhance the classification performance for high dimensional datasets, and the datasets that we used in this thesis. Chapter 5 includes the the experiments and the results of our approach. Finally, Chapter 6, concludes the thesis and propose some new directions for future work perspectives.

Chapter 2

Background

This chapter presents a background and the theoretical concepts that are mainly related to this thesis. First, we introduce the machine learning concept. Then, we will talk about data classification and the classifiers used in our thesis. The next subsection will be about general hierarchies for parallel evolutionary algorithms. The rest of this chapter presents the SS algorithm and parallel SS that are related to this work.

2.1 Machine learning

Machine learning (ML) "is a sub-field of *artificial intelligence* (AI), that provides computers with the ability to learn without being explicitly programmed" [12, 13].

Machine learning algorithms are often categorized as being supervised, unsupervised, or reinforcement learning [1, 8, 12], in the following we clarify these types:

- Supervised learning: is the machine learning task of inferring a function from labeled training data. This type of ML can be divided into two types:
  - Classification; where each data sample in training dataset is a pair consisting of an input vector and its desired output value (class). Classification the out put is discrete.
  - Regression learning, where the output is continuous.
- Unsupervised learning: this type of learning in which the input is an unlabeled training samples, where, the task is to find hidden structure in unlabeled dataset (cluster the data to discrete outputs or classes).

- Reinforcement learning: this type is inspired by behaviorist psychology, concerned with how software agents must take actions in an environment so as to maximize some notion of cumulative reward.

2.2 Data classification

Classification in supervised machine learning, "it is the task of predicting the class membership of categorized elements; whose classes is not known. using the properties of examples in a model learned previously from training examples; whose classes was known" [14].

Classification has two phases, training phase and prediction phase. In the training phase, the model tries to understand the relation between the input data and the output data (classes). While in the prediction phase, the model predict the classes of the categorized elements [15, 16]. In this thesis we used two classifiers which are Support Vector Machine (SVM) and Random Forest (RF), in the following subsections we will explain each of them.

2.2.1 Support Vector Machine classifier

SVM is a binary discriminative classifier which tries to find the optimal hyper plane that efficiently separates nearest data points from different classes. The main goal of SVM is to draw a hyper plane that classifies all data from two different classes. This hyper plane maximizes the margin from both classes. If we suppose that hyper plane is represented by equation 2.1 [1], then the total margin is represented by $\frac{2}{\|\vec{w}\|}$. Let $\{X_1, X_2, ..., X_n\}$ is dataset and $y_i \in [-1, 1]$ is the class label for this dataset. Then the decision boundary that should classify all points correctly is represented by equation 2.2 [8]. So we should minimize $\frac{1}{2}\|w\|^2$ to maximize the margin, this minimizing is a nonlinear optimization task solved by Karush Kuhn Tucker (KKT) condition [1, 8].

$$g(\vec{x}) = \vec{w}^T \vec{X} + \vec{w}_0 \tag{2.1}$$

$$y_i(\vec{w}^T X_i + b) \geq 1 \tag{2.2}$$

2.2.2 Random Forest classifier

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [17]. Random Forests are an ensemble of learning methods for classification, regression and other tasks, it operates by constructing a multitude of decision trees at training time and outputting the class that is the classes (classification) or mean prediction (regression) of the individual trees, Figure 2.1 explains the RF Algorithm Steps [1, 8].

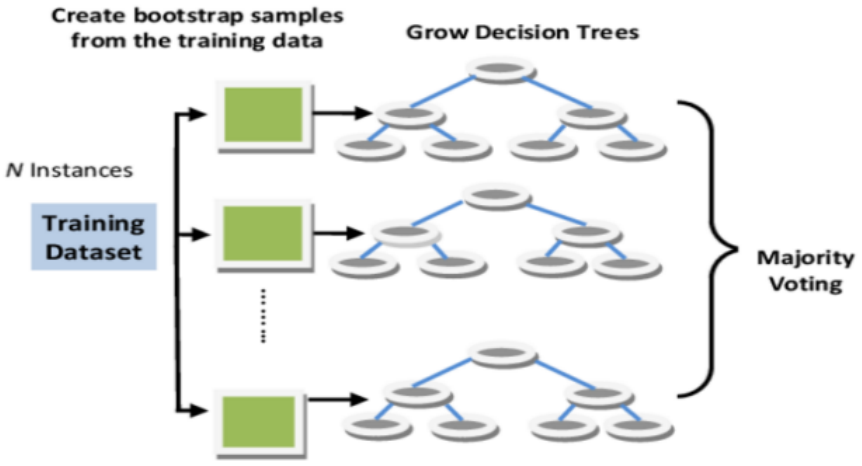

FIGURE 2.1: Random Forest algorithm steps

Randomly the dataset is sampled and replaced with bagging or bootstrap aggregation, in addition of the identical growing of distributed trees; thus the anticipation is the same for both the averaged trees and the individual trees [1].

The advantages are achieved from the averaging process are results by the amount of interconnection of the bagged trees pairs. For further variance reduction, specifically at the growing stage of trees; each node split of each tree could be undergoing a random elimination step for the number of features inside it, this minimizes the correlation between trees which are in turn and reduce the variance [1].

An important improvement is considered for RF when building the forest by disassembling the trees, which is that a class is being predicted by a vote of each tree commission, the new x vector has input features which have been labeled or y by the majority class [1, 8].

2.3 Dimension reduction

In general, there are two main types of data reduction models [9]. The first one is instance selection and instance generation processes are focused on the instance level. (i.e. select a representative portion of data that can fulfill a data mining task as if the whole data is used) [18]. The second is feature selection and feature extraction models work at the level of characteristics.

Features in any datasets can be divided into three categories: relevant, irrelevant, and redundant features [9, 19]. The relevant features, are the features that have an impact on the output and their role can't be replaced from any other features. While, the irrelevant features don't have direct impact on the output, but they are affecting the learning process. The last one is the redundant features are those ones that another feature can take their role and they don't have direct impact on the target concept.

In dimension reduction, we transform high-dimension data into a meaningful representation of reduced dimensionality. In addition, the input variables subset can be defined through two different approaches: feature selection and feature extraction [20, 21].

Dimension reduction has many advantages, the most important one that it can facilitate classification; since it reduces complexity and gives simpler data representation. Dimension reduction models attempt to reduce a dataset by removing noisy, irrelevant, or redundant features. As a result, the feature selection is needful preprocessing step in analyzing high dimensional datasets. It often leads to smaller data that will make the classifier training better and faster [22]. In this section, we will explain feature selection and feature reduction.

2.3.1 Feature extraction

In feature extraction, the dimensionality of features reduced but the variant information maintained. The new extracted features are a combination of original features. The most popular feature extraction techniques are: *Principle Component Analysis* (PCA), *Linear Discriminative Analysis* (LDA), and *Canonical Correlation Analysis* (CCA) [8, 23]:

- *PCA*: it is a popular unsupervised dimension reduction technique. PCA is a linear transformation that transforms data to a new set of variables which called principle components (PCs), by referring it to different coordinate system. These PCs are uncorrelated to each other, which minimize the redundancy. The PCs are ordered based on their importance, then the dimension reduction is done by eliminating the PCs that don't have significant importance.

- *LDA*: it is a popular supervised technique; this approach maximizes the distances between classes and minimizes the distances between the samples in each class. This approach is a parametric method; the discriminatory information must be in the mean range values and not in the variance of data.
- *CCA*: this approach is used for correlating linear relationships between two multi-dimensional variables.

2.3.2 Feature selection

Feature selection is considered to be a complex and time-consuming problem especially with high-dimensional datasets [8]. In order to make classification faster and increasing its performance, we need to select the subset of features that are discriminative. Searching in the feature space has a high computational cost. For instance, if there are D features in the search space, then the total search space is (2^D) [22, 23]. This means that this problem is NP-hard for large value of D.

In general, there are three classes of feature selection approaches: filter-based, wrapper, and embedded [1]. The filter approach analyzes the features statistically and ignores the classifier. Most of these methods perform two operations, ranking and subset selection. In some cases, these two operations are performed sequentially, first the ranking, then the selection, in other cases only the selection is carried out. These methods are effective in terms of execution time. However, filter methods sometimes select redundant variables; since they don't consider the relationships between variables. Therefore, they are mainly used as a pre-processing method.

In the wrapper model [24], the process of feature selection is depending on the performance of a specific classifier. But its disadvantage is long execution time. The last method for feature selection is the embedded. In this method, the feature selection process and the learning algorithm (tuning the parameters) are combined [24].

Based on the fact the wrapper model is more accurate than the filter model; in this study the wrapper approach of feature selection is used. In the following subsection we detail with this model.

2.3.3 Wrapper model

In the wrapper model, the process of feature selection is based on the performance of a specific classification model [24]. This means that the performance of wrapper model is higher than it in the filter approach. But, computational learning time is higher [1].

As seen in Figure 2.2 [8], The main steps of wrapper model are [24]:-

- Generating a subset of features.
- Evaluating the selected subset of features by using the performance of the classifier.
- Repeating step1 and step2 until the desired quality is reached.

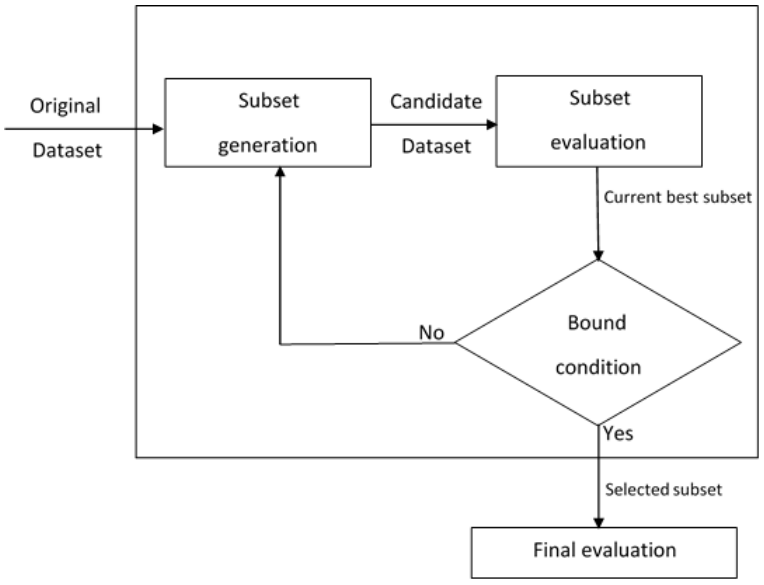

FIGURE 2.2: General wrapper model for feature selection

The feature selection in wrapper model as black box; the classifier is used to evaluate the subset of features based on their prediction performance. Then the subset of features of the highest performance will be used as the final set to learn the final classifier [1, 8].

The advantages of wrapper model are [24]:-

- Accuracy: this approach has high accuracy; since it tuned the specific interaction between the classifier and the dataset.
- It has a mechanism to avoid over-fitting; because it uses cross-validation in accuracy prediction.

The disadvantages of wrapper model are [1, 8, 24]:-

- The execution time is high for high dimensional dataset.
- Less general: It depends on the classifier of selecting the best subset of features.
- It may be selected the features that are biased to the classifier it used.

2.4 Scatter search algorithm

Scatter search is a *Metaheuristic* and a *Global Optimization* algorithm. It is also sometimes associated with the field of *Evolutionary Computing* given the use of a population and recombination in the structure of the technique. Scatter Search is a sibling of Tabu Search, developed by the same author and based on similar origins [25, 26].

Scatter search algorithm is an evolutionary Algorithm that was successfully applied to hard optimization problems. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems [27].

The main steps of scatter search algorithm are shown in algorithm 1. As illustrated in Figure 2.3 [25], the implementation of SS algorithm to find the best solution for any problem has five steps as follow [5, 28–30]:-

- Diversification initialization: in this step, we generalize the initial population of solutions. The first part of this population is generated randomly, then the rest solutions are the diverse solutions.
- Solution improvement: this step takes the most execution time of this algorithm. In this step, we try to find the best solution of each solution in the initial population.
- Generate Reference Set: after the improvement, we choose the subset of the best solutions and a subset of the worst solutions in the population to add them to the Reference Set.
- Solutions combination: the solutions in the Reference set are combined to generate new solutions.
- Update the Reference Set: in this step, we choose the best combined solutions and add them to the Reference Set.

2.5 Parallel scatter search algorithm

The *Parallel Scatter Search Algorithm* is an extension of the SS algorithm. The main motivation of parallel SS is to reduce the execution time of SS which is needed to reach an acceptable solution [5, 31].

In general there are two main approaches of parallel SS, the standard and the decomposition [31]. In the standard parallel SS, the algorithm is applied on one population

Algorithm 1 Sequential scatter search methodology [5]

```
Create Population (Pop, PopSize)
Generate ReferenceSet (RefSet, RefSetSize)
while Not Stopping Criterion1 do
  while Not Stopping Criterion2 do
    Select Subset (Subset, SubsetSize)
    Combine Solutions (SubSet, CurSol)
    Improve Solution (CurSol, ImpSol)
  end while
  Update ReferenceSet (RefSet)
end while
```

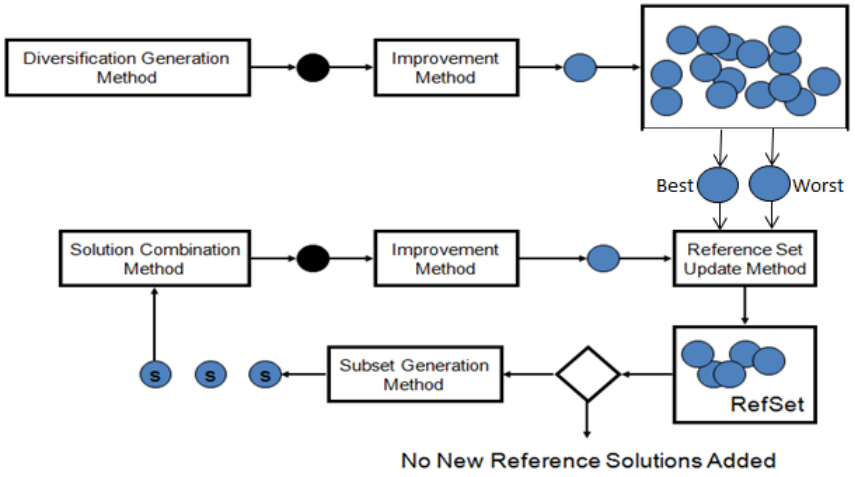

FIGURE 2.3: Scatter search algorithm

only, such as master-worker architecture. While, in the decomposition parallel SS the population is divided into sub populations, then the algorithm is applied on each sub population in parallel, such as distributed architecture. The improvement and iterations in both approaches are done in parallel.

There are three main architectures for parallel evolutionary algorithms, which are: master-worker, cellular, and island architecture [2, 31].

- Master-worker architecture: also known as global parallelization or framing, it uses centralized control. In this parallel architecture, the master is the central processor that initialize the population. After initialization, each worker processor receives a subset of solutions from the worker to evaluate the fitness of each one. After the workers finished their works, they return the fitness values to the master. When the master receives the evaluation results form the worker it performs the rest of operations on the population. Figure 2.4 shows the master-worker architecture.

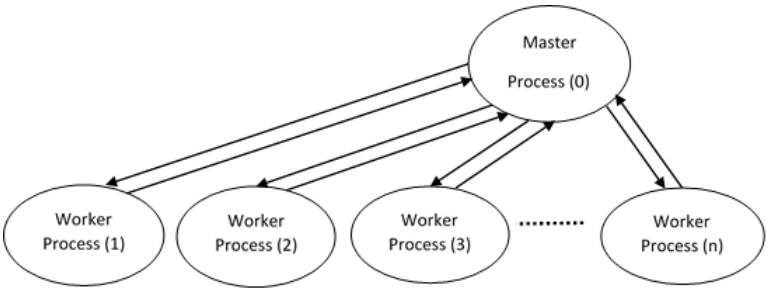

FIGURE 2.4: Master-Worker parallel architecture

- Cellular architecture: also known as coarse-grain, it uses distributed control. In this architecture, the concept of neighborhood is used, this means that the individual may only interact with its nearby neighbors in the breeding loop. This interaction in order to find a good solution across the overlapped individuals.
- Island or distributed architecture:in this parallel architecture, the population is divided into sub-populations (islands) in which isolated serial algorithms are executed. There is individuals exchange between these islands; in order to keep the diversity in the sub-population.

Furthermore, there is a hybrid model in which two different parallel architectures may be used [2]. In general, the higher level is a coarse-grain (set of islands) parallel architecture, then each island perform other parallel architecture from the three mentioned architectures.

In this theses we used the master-worker architecture to build the parallel SS algorithm, the master process initializes the population of solutions, then it distributes these initial solutions equally among the worker processes for improvement and evaluation. Then the master collect the solutions from workers to complete the SS operations on the improved solutions. The master-worker architecture for our approach will be explained in details in Chapter 4.

Chapter 3

Literature review

This chapter is a bout previous works that are dealing with feature selection problem in high dimensional datasets. It is organized as follow: the first Section will be about parallel programming in general. The next Section is an overview of parallel evolutionary algorithms. The third Section talks about the previous works in parallel evolutionary algorithms for feature selection in high dimensional datasets.

3.1 Parallel programming

Parallel programming is the running of many parts of the task at the same time on different processors. Nowadays, the most popular systems have multicore. In order to get the benefit from these systems we need to use parallel programming [32, 33].

Parallel programming explicitly breaks the task down into many processes, where each processes can be executed in parallel on a single processor. In this way multiple parts of the same task can run in parallel at the same time [33].

3.2 Parallel evolutionary algorithms for feature selection

Feature selection algorithms are used to find an optimal subset of relevant features in the data [10, 34]. In this section we will talk about parallel evolutionary algorithms that are used for feature selection problem in high dimensional datasets. In the following sections, we will clarify the steps of five parallel evolutionary algorithms.

3.2.1 Parallel genetic algorithm

In order to increase the efficiency and reduce the execution time of the genetic algorithm (GA); the researchers used parallel GA. Algorithm 2 presents the parallel GA methodology, with the master-slave model of parallel GA.

Algorithm 2 Parallel genetic algorithm [35]

```
Create initial population
Evaluate initial population
Create slaves
while not done do
  Start slave
  Wait for slave to finish
  Run mutation operator
end while
for i=1 to slave iterations do
  Select individuals
  Run crossover operator
  Evaluate offsprings
  if solution found then
    set done=True
  end if
end for
```

3.2.2 Parallel CHC algorithm

A CHC is a non-traditional GA, which combines a conservative selection strategy (that always preserves the best individuals found so far), that produces offsprings that are at the maximum hamming distance from their parent. The main processes of CHC algorithm are [9]:

- Half-Uniform Crossover (HUX): This will produce two offsprings, which are maximally different from their two parents.
- Elitist selection: this will keep the best solutions in each generation.
- Incest prevention: this step prevents two individuals to mate if the similarity between them greater than a threshold.
- The Restarting process: if the specified population stagnated, then this step generated a new population by choosing the best individuals.

3.2.3 Particle swarm optimization (PSO)

This subsection handles the geometric particle swarm optimization (GPSO) and shows the algorithm that used to parallelize PSO or GPSO.

3.2.3.1 Geometric particle swarm optimization (GPSO)

GPSO is a recent version of PSO. The key issue in GPSO is the using a multi-parental recombination of solutions (particles). In the first phase, a random initialization of particles created. Then the algorithm evaluates these particles to update the historical and social positions. Finally, the three parents (3PMBCX) move the particles [36], as shown in Algorithm 3:

Algorithm 3 GPSO algorithm [4]

```
S:SwarmInitialization()
while not stop condition do do
  for each particle i of the swarm S do do
    evaluate(solution(xi))
    update(velocity equation (hi))
    update(global best solution (gi))
  end for
  for each particle i of the swarm S do do
    xi:3PMBCX ((xi, wa), (gi, wb), (hi, wc))
    mutate(xi)
  end for
end while
Output: best solution found
```

3.2.3.2 Parallel multi swarm optimization

Parallel multi swarm optimization presented in [4], it was defined in analogy with parallel GA as a pair of (S, M), where S is a collection swarm, and M is a migration policy. Algorithm 4 depicts the parallel PSO methodology.

3.2.4 Parallel scatter search

Scatter search is an evolutionary method that was successfully applied to hard optimization problems. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems, see Algorithm 5.

Algorithm 4 Multi swarm optimization [4]

```
DO IN PARALLEL for each i  1,...,m
initialize(Si)
while not stop condition do do
  iterate Si for n steps /* PSO evolution */
  for each Sj  (Si) do  do
    send  particles in s(Si) to Sj
  end for
  for each Sj such that Si  (Sj ) do do
    receive  particles from Sj
    replace  particles in Si according to r
  end for
end while
Output: best solution ever found in the multi-swarm
```

Algorithm 5 Parallel scatter search methodology [5]

```
Create Population (Pop, PopSize)
Generate ReferenceSet (RefSet, RefSetSize)
while  Stopping Criterion1 do
  while  Stopping Criterion2 do
    Select Subset (Subset, SubsetSize)
    for each processor r=1 to n do in parallel do
      Combine Solutions (SubSet, CurSol)
      Improve Solution (CurSol, ImpSol)
    end for
  end while
  Update ReferenceSet (RefSet)
end while
```

3.2.5 Parallel ant colony optimization

When dealing with huge search space, parallel computing techniques usually applied to improve the efficiency. Parallel ACO algorithms can achieve high-quality solutions in reasonable execution times comparing with sequential ACO. In Algorithm 6, the methodology of PACO is presented.

3.3 Related work

We reviewed a set of research papers, which were dealing with feature selection problem for high dimensional datasets in a parallel environment and using parallel evolutionary algorithms. Let us discuss these studies in the following subsections.

Algorithm 6 Parallel ant colony optimization methodology [37]

```
Generate Ants
Initialize N processors
Multicast to all slaves processors N and the task ids of all slaves
for each slave do do
  Send a number between 0 and N that identifies the task inside the program
end for
while not all slaves have sent back solution do
  Wait for solution
  if a slave returns a solution that is better than any solution received then
    Multicast this solution to all slaves
  end if
end while
Return the best solution
```

3.3.1 Parallel GA

Liu et al., [11] used parallel GA for selecting informative genes (features) in tissue classification, using wrapper approach. The main purpose was to find the subset of features with fewer elements and higher accuracy. The parallelization of GA performed by dividing the population into sub-populations, and then the GA run on each sub-population. Therefore, the searching for the optimal subset of genes can be on several CPUs/computers at the same time.

For evaluation, the Golub classifier was used. This classifier introduced by the authors and it depend on the sign of the results for classification; if the sign is positive the sample x belongs to class 1, else if it negative the sample x belongs to class 2. This classifier used only if the datasets have two classes. The accuracy of the classifier tested by using the LOOCV (leave one out cross validation) method. The results showed that using the parallel GA increased the accuracy, and reduced the number of genes that used for classification.

In [38] Zheng et al., analyzed the execution speed and solution quality of many parallel GA schemes theoretically. Furthermore, they pointed to the best scheme of parallel GA that used on multi-core architecture. This paper considered the relationship between speed and parallel architecture along with solution quality.

They analyzed (Master-Slave, Synchronous Island, Asynchronous Island, Cellular, and hybrid scheme of Master-Slave and Island) schemes of parallel GA, with Pthread library on multi-core parallel architecture.

To validate their theoretical analyzing an experiments performed. The hybrid scheme of (Master-Slave and Asynchronous Island) was the best scheme in performance using multi-core architecture. The Island scheme has the best execution time, but the worst

solution quality. To improve the solution quality when using Island model it is better to decrease the number of islands. The Asynchronous Island is faster than the Synchronous. The Master-Slave scheme has the best solution quality and the worst execution time.

Soufan at el., [24] developed a web-based tool called DWFS, which used for feature selection for different problems. This tool followed a hybrid approach of wrapper and filter. First, the filter used as preprocessing and select the top ranked features based on tunable and a predefined threshold. In the next step, parallel GA based on wrapper approach applied to the selected features to search for subset features that increase the classifier accuracy. The scheme of parallel GA was Master-Slave; the master node used to create initial population and GA steps. While the slave (worker) nodes used for fitness evaluation of each chromosome, this implementation is performed on 64 core.

For evaluation, they used three different classifiers (Bayesian classifier, K-nearest neighbor, and a combination of them). The results of the experiments show that DWFS tool provided many options to enhance the feature selection problem in different biological and biomedical problems.

In [39] Pinho et al., presented a framework called ParJEColi (java-based library) for a parallel evolutionary algorithm in bioinformatics applications. The aim of this platform was to make the parallel environment (multi-core, cluster, and grid) easy and transparent to the users. This library adapted itself to the problem and the target parallel architecture. The user can easily configure the parallel model and the target architecture; since, ParJEColi encapsulated the parallelization concerns as features. The explicit steps implemented by a simple GUI.

The experiments for validation this framework were done on 2 biological dataset and many bioinformatics scenarios. The results indicate that the proposed framework improves the computational performance (decreases execution time) also the solution quality.

3.3.2 Parallel CHC

In [9] Peralta et al., presented a parallel evolutionary algorithm called CHC algorithm by using the MapReduce paradigm for selecting features in high dimensional datasets to improve the classification. The parallelization of CHC algorithm is done by using MapReduce procedure (Hadoop implementation).

A cluster of computers of 20 computing nodes were used. Each dataset split into 512-map task. For evaluating their work, three classifiers where used SVM (support vector machine), logistic regression, and Bayesian classifier.

The results showed that the run time for classification increased as the number of features decreased, except for Bayesian classifier. They explained this result as follow: if the number of blocks less than the number of computing machines; this leads to have some machines remain idle. In addition, if the number of blocks greater than the number of computing machines, the blocks maybe will not distributed in efficient way.

They compared parallel CHC with the serial version, and they concluded that the accuracy of classification increased by using parallel CHC. Furthermore, the parallel version of CHC reduced the run time when the datasets is high dimensional.

3.3.3 Parallel PSO

PSO is an efficient optimization technique, it used to solve the problem of feature selection in high dimensional datasets. In [20] Chen et al., used the parallel PSO algorithm for solving two problems at the same time. By creating an objective function that takes into account three variables at the same time (the selected features, the number of support vectors, and average accuracy of SVM). In order to maximize the capability of SVM classifier in generalization.

The proposed method called PTVPSO-SVM (parallel time variant particle swarm optimization support vector machine), it had two phase: 1) the parameter settings of SVM and feature selection work together. 2) the accuracy of SVM evaluated using the set of features and the optimal parameters from the first phase.

They used parallel virtual machine (PVM) with 8 machines; and 10-fold cross validation. The results showed that they could achieve the following aims: increasing the accuracy classification, reducing the execution time comparing with sequential PSO, producing an appropriate model of parameters, and selecting the most discriminative subset of features.

Feature selection can be carried out based on rough set theory with searching algorithm as in [10, 22].

In [10] Qian et al., proposed three parallel attribute reduction (feature selection) algorithms based on MapReduce on Hadoop. The first algorithm was built by constructing the proper (key, value) by rough set theory and implementing MapReduce functions. The second algorithms were done by realizing the parallel computation of equivalence classes and attribute significances. The last parallel algorithm was designed to acquire the core attributes and a reduce in both data and parallel task.

The experiments are performed on a cluster of computers (17 computing node). They considered the performance of the parallel algorithms, but they did not focus on the

classification accuracy; since the sequential and parallel algorithms gave the same results. The results showed that the proposed parallel attribute reduction algorithms could deal with high dimensional datasets in an efficient way and better than the sequential algorithms.

In [22] Adamczyk, use rough set theory for attribute reduction, to increase the efficiency he implemented parallel Asynchronous PSO for this problem. The parallelization was done by assigning the complex function computations in slave cores and the main core make the updating particle and checking the convergence of the algorithm.

From their experiments it was noticeable that the efficiency and speedup of parallel PSO algorithm were raising as the size of dataset increased. The achievable accuracy was not astonishing, but it was better than the classical algorithms.

3.3.4 Parallel GPSO

In [4] Garcia-Nieto et al., parallelized a version of PSO called GPSO which is suitable for feature selection problem in high dimensional datasets. The proposed method was called PMOS (Parallel multi-swarm optimizer). Which was done by running a set of parallel sub PSOs algorithms, which forming an island model. Migration operation exchanged solutions between islands based on a certain frequency The aim of the fitness function increasing the classification accuracy and reduce the number of selected genes (features).

They used the SVM classifier (Support Vector Machine) to prove the accuracy of the selected subset of features. In their experiments, they used a cluster of computers as a parallel architecture. They found that 8-swarm PMSO was the best choice for parallelization. The results pointed out that this algorithm was better than the sequential version and other methods in term of performance and accuracy while it selected few genes for each subset.

3.3.5 Parallel SS

In [5] Lopez et al., present a parallel SS metaheuristics for solving feature selection problem in classification. They proposed two methods for combining solutions in SS.
The first method is called GC (greedy combination): in this strategy, the common features of the combined solutions are added, then at each iteration one of the remaining features is added to any new solution.

The second strategy is called RGC (reduced greedy combination), it has the same start as GC, but in the next step, it considers only the features that appear in solutions with

good quality. Then the parallelization of SS is obtained by running these two methods (GC, RGC) at the same time on two processors. Using different combination methods and parameters settings at each processor.

They compared the proposed parallel SS with sequential SS and GA. The results show that the quality of solution in parallel SS is better than solutions which was obtained from the sequential SS and GA. Also, the parallel SS use a smaller set of features for classification. The run time is the same for parallel and sequential SS.

3.3.6 Parallel ACO

This subsection shows how the parallel ACO is used to solve feature selection problem for classification in high dimensional datasets.

In [40] Meena et al., implemented a parallel ACO to solve the feature selection problem for long documents. The parallelization was done using MapReduce programming model (Hadoop) that automatically parallelize the code and data then run them on a cluster of computing nodes. The wrapper approachis used as evaluation criteria that used Bayesian classifier. Furthermore, the accuracy of the classifier was based on these metrics: precision, recall, accuracy and F-measure.

The enhanced algorithm (parallel ACO) was compared with ACO, enhanced ACO, and two feature selection methods, CHI (Statistical technique) and IG (Information Gain). They used Bayesian classifier in evaluation process. The results showed that for a given fixed quality of the solutions the proposed algorithm could reduce the execution time but without considered the solution quality. On the other hand, the accuracy of the classifier was increased using parallel ACO comparing with sequential ACO and feature selection methods.

In [14] Cano et al., parallelized an existing multi-objective ant programming model that used as the classifier. This algorithm was used for rule mining in high dimensional datasets. The parallelization was done on data and each ant encoded a rule. This was achieved by let each processor perform the same task on a different subset of the data at the same time. In the implementation, they used GPUs, which are multi-core and parallel processor units architecture. This parallel model Followed CUDA method.

For evaluation they used these metrics: true positive, false positive, true negative, false negative, sensitivity, and specificity. The results indicate that the efficiency of this model was increased as the size of datasets increased.

TABLE 3.1: Summary of algorithms and programming models

Algorithm	**Used evolutionary algorithm**	**Parallel Programming model**
Peralta et al., [9]	CHC (Version of GA)	MapReduce
Garcia-Nieto et al., [4]	GPSO	MALLBA
Adamczyk [22]	PSO	Unknown
Chen et al., [20]	PSO	PVM
Liu et al., [11]	GA	Unknown
Lopez et al [5]	SS	Unknown
Soufan et al., [24]	GA	MPI
Meena et al., [40]	ACO	MapReduce

3.4 Summary

The summary of the papers that implemented the parallel EA for solving the classification problem in high dimensional datasets is reported in Table 3.1 and Table 3.2.

Many research papers [4, 14, 22, 35, 38, 39, 41], stated that we can reduce the execution time and achieve acceptable speed ups, when applying parallel evolutionary algorithms on multiple processors. We noticed that they achieved a reasonable speed up in many cases.

In the next table (Table 3.2), when comparing the accuracy of parallel EA it is important to notice how many classifiers were used to measure the accuracy. Furthermore, we should consider the metrics that were used to evaluate the classifier. For example, the parallel PSO and its variants have the higher accuracy; but they used only one metric which is the success rate. This means that the parallel PSO is not the most accurate parallel EA based on Table 3.2.

On the other hand, the parallel GA and its variant has the least accuracy, but they used from two to five metrics for evaluation purpose. Based on these metrics, we can say that the parallel GA is the best parallel EA for feature selection in high dimensional datasets.

After the review of different parallel EA that are used to solve the feature selection problem in high dimensional datasets. We adopted the accuracy as a measure to compare the algorithms performance.

TABLE 3.2: Summary of algorithms, datasets, classifiers, and classification results

Algorithm	**Dataset**	**Classifier**	**Metric for Classification**	**Result**
Peralta et al., [9]	Epsilon	Bayesian	AUC = (TPR+TNR)/2	0.17
		SVM		0.68
		Logistic Regression		0.70
	ECBDL 14 ROS	Bayesian		0.67
		SVM		0.63
		Logistic Regression		0.63
Nieto et al., [4]	Colon	SVM	Success Rate	0.85
	Lymp			0.97
	Leuk			0.98
	Lung			0.97
Adamczyk et al., [22]	15 datasets	—-	Success Rate	0.70 (Avg)
Chen et al., [20]	30 datasets	SVM	Success Rate	0.87 (Avg)
Liu et al., [11]	Leukemia	Golub	Success Rate	0.88
	Colon			—-
Lopez et al., [5]	12 datasets	Nearest Neighbor	Success Rate	0.86 (Avg)
		Bayesian		0.87 (Avg)
		Decision Tree		0.86 (Avg)
Soufan et al., [24]	9 datasets	K Nearest Neighbor	GMean	0.81 (Avg)
		Bayesian		0.79 (Avg)
Meena et al., [40]	2 datasets	Bayesian	Recall Precision	0.64 (Avg)

The following points show our conclusion about the performance of the mentioned algorithms in this chapter for feature selection:

- GA and its variants: based on the papers we reviewed, the parallel GA has the higher accuracy.
- PSO and its variants: the parallel PSO has the same accuracy as sequential PSO.
- SS: parallel SS gives better results in case of accuracy than GA and sequential SS.
- ACO: parallel ACO has the less accurate results than the other parallel EA.

Chapter 4

Proposed algorithm and methodology

In this chapter we will explain the benchmark datasets which have been chosen in this research. The methodology that we used in this thesis aims to enhance the classification accuracy and reduce the execution time, especially when dealing with high dimensional datasets. The proposed approach is used the wrapper feature selection method based on scatter search algorithm.

The first Section 4.1, will introduce the benchmark datasets. In Section 4.2, we will show a general description of our approach. Then, in Section 4.3, we will talk about the sequential version of the proposed approach. Finally, Section 4.4, will handle the parallel version of the proposed approach.

4.1 Benchmark datasets

Five benchmark datasets (Arcene, Madelon, Mushroom, Spambase, and Gisette) have been chosen in order to test our proposed approach. All datasets were used in this work are two-class classification problems. Arcene, Madelon, and Gisette datasets are considered as feature selection challenge [42].

In order to compare our proposed wrapper approach built on sequential scatter search with wrapper approach that built on genetic algorithm [8], and with filter approach built on particle swarm optimization algorithm [1]; we used the same datasets which are: Madelon, Arcene, and Mushroom datasets. In our experiments we used the same virtual server that was used in [1] and [8].

TABLE 4.1: Summary of the datasets used in the experiments

Dataset	Number of features	Samples in Training data	Samples in Testing data	Samples in Hidden data	All samples
Madelon	500	1334	666	600	2600
Arcene	10000	67	33	100	200
Mushroom	22	3611	1805	2708	8124
Spambase	57	2046	1022	1533	4601
Gisette	5000	4000	2000	1000	7000

The spambase dataset used in order to compare our proposed approach with other works like [43], in which they are used the neural networks for classification. The Gisette dataset used in order to compare the proposed approach with other works like [44], in which they are used parallel implementation in R language of the weighted subspace random forest algorithm.

All the datasets that we used in this work are taken from UCI machine learning repository [42]. Table 4.1 shows a summary of these datasets, the number of features, and the number of samples. The samples in each dataset are divided between three parts: training data, testing data and hidden data. The last part which is the hidden data is used for final evaluation of the selected subset of features.

4.2 General description of the proposed approach

This section will discuss the the wrapper steps of the proposed approach, then the SS operations will be described, after that we will talk about the cross validation that is used in this research, finally, the performance evaluation and fitness calculation.

Theoretically, the best subset of discriminate features can be done by finding all possible candidates for the solution and checking whether each candidate satisfies the problem's statement, which known as exhaustive search. But in order to perform it on the feature space, it needs to search 2^n possible subsets of n features; this is almost impractical to conduct it [8]. On the other hand, metaheuristics methods is suitable way to find solution for this problem in a reasonable time [4, 6, 7]. This motivate us to develop our approach to handle the feature selection problem in high dimensional datasets.

The proposed approach is built on wrapper approach that uses scatter search algorithm for feature selection. This research aims to enhance the classification performance and reduce execution time especially for high dimensional datasets. However, these advantages come with a cost, as searching for relevant and discriminative feature subset that increase the classification accuracy introduces additional computational complexity and requires a high dimensional number of fitness evaluations to get an adequate solution.

4.2.1 Wrapper feature selection steps in the proposed approach

The wrapper approach relies on a classification algorithm that is used as a black box to evaluate each candidate subset of features. In this work the algorithm is SS, the major bottleneck for SS feature selection is the classifier training; because the training phase is repeated a lot of times in every step of the evolutionary process.

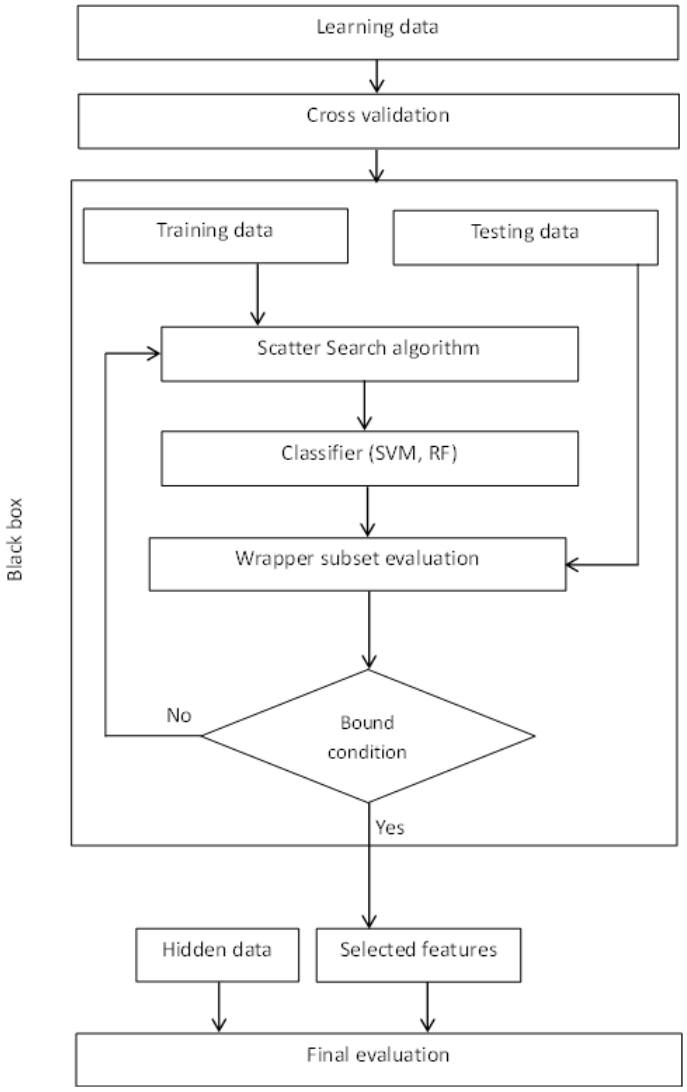

FIGURE 4.1: Wrapper feature selection using SS algorithm

As seen in Figure 4.1, the datasets is split into learning data and hidden data. The learning data is split into two part: training and testing. The training data is used to train the model black box, the testing data is used evaluate the candidate subset of selected features. The hidden dataset is used in final evaluation for the best subset of the selected features.

The steps of the wrapper feature selection model that is used in this research are as follow:-

1. A candidate subsets of features are generated by using a search strategy, in our approach we used the Scatter Search algorithm.

2. Evaluate the generated candidate subsets of features using wrapper subset evaluater. The quality of the candidate subset is evaluated by the performance of the chosen classifier obtained on the training data.

   This step is continues until reaching the bound condition. There are two types of bound conditions which are [8]:-

   - Bound condition based on a generation procedure: in this type the number of iterations or the number of selected features is predefined.
   - Bound condition based on an evaluation function: the operations are repeated until optimal subset is reached according to some evaluation function.

   In this research the first type is used since; the number of iterations is predefined. Our experiments show that for this type of search algorithm (SS), few numbers of iterations will return the final result within a reasonable time. So, the range of iterations used in our experiments is **(3-9)**

3. Validation; where we check whether the selected feature subset is valid or not based on the classification results.

As mentioned in the first step of the wrapper feature selection model, we used a search strategy in order to generate a subsets of selected features. In this work, we used the scatter search algorithm for this purpose. There are two versions of the SS that have been implemented in this research, which are sequential scatter search, and parallel scatter search algorithms. The next sections 4.3, and 4.4 will discuss these two versions of the SS.

4.2.2 Scatter search algorithm operations

In our research, we have customized some operations of the SS in order to enhance the classification accuracy and reduce execution time. First of all, we will talk about the solution representation that is used in our algorithm, then the operations of SS will be explained.

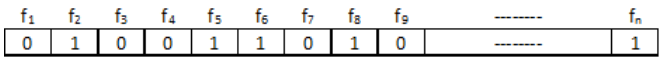

FIGURE 4.2: Solution representation in the proposed approach

Solution representation:-

In our SS algorithm the representation of the solution used is illustrated in Figure 4.2. Each solution S is a vector of length n, where n is the number of features in the dataset. If the feature f is selected the value that represent this feature will be 1, while 0 means that this feature is not selected in this solution. The solution can be defined as: $S = \{f_0, f_1, ..., f_n\}$.

The operations of SS algorithm are:-

- Generating the initial population of solutions: in this operation we generate the initial set of solutions (Pop) based on a predefined population size. The size of the initial population ($PSize$) of solutions is ranged from **8** to **64** solutions. This population is divided into two equals sets. The first set of solutions is generated randomly. The second set of solutions is generated by inverting each solution in the first set. We use this technique in the second set of solutions to guarantee the diversity in the population. The initial population of solutions can be defined as: $pop = \{S_{11}, S_{12}, ..., S_{1n}\}$, where n is the number of solutions in the population.

$$S_{ij} = \begin{cases} f_{i0}, f_{i1}, ..., f_{in} & i < \frac{PSize}{2} \\ \bar{f_{i0}}, \bar{f_{i1}}, ..., \bar{f_{in}} & i \geq \frac{PSize}{2} \end{cases}$$

  This operation can be understood from the example that is illustrated in Figure 4.3. For example f_1 in the first solution is not selected (0), this means that this feature will be selected (1) in the second solution.

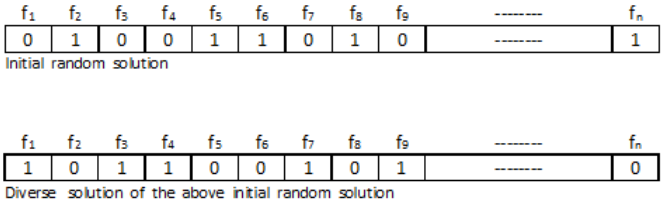

FIGURE 4.3: Generating an initial random solution and its diverse solution

- Improvement operation: in our work we implemented the improvement operation in two different ways: deterministic improvement and random improvement, the descriptions of them are as the following:-
  - In the deterministic improvement operation: we generate n candidate solutions from each initial solution, where n is the number of features in the

dataset. Then after evaluating all these solution we will choose the best solution from the *n+1* solutions (*n* candidate solutions and one initial solution).

This way is illustrated in Figure 4.4 , this example explains how the candidate solutions are generated from the solution. By changing the value of each feature we will have a new solution. Here there are 9 features, so we will generate 9 candidate solutions from this initial solution. Since it has 9 features, and each time the value of each feature will be changed. For example, the first feature f_1 is not selected (0) in the initial solution, so it will be selected (1) in the first candidate solution.

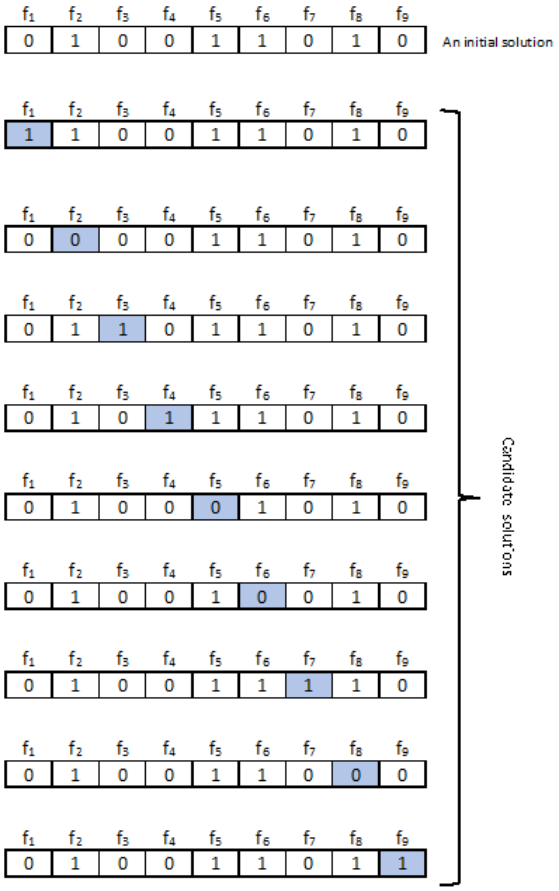

FIGURE 4.4: Deterministic improvement operation

– In the random improvement operation: the number of generated candidate solutions from each initial solution will be a random number of solutions that ranged from **5** to **n**, where **n** is the number of the features in the dataset. Furthermore, the number of changed values for the features is a random number that ranges from **1** to $n/5$, this number **(n/5)** is chosen according to

our experiments. Then after evaluating all these candidate solution we will choose the best solution.

In Figure 4.5 we show an example that helps in understanding the second way of the random improvement operation that is used in the proposed approach. In this example, r is a random number, this number determines how many features we will change their values. For example, at the first time r=1 this means that the value of the first feature f_1 will be changed from not selected (0) to selected (1). But, in the second time r=3 this means that the values of (f_2, f_3, and f_4) will be changed from (1, 0, 0) to (0, 1, 1) respectively.

The number of candidate solutions that are obtained from the random improvement operation is C:

1. Maximum C: is obtained from minimum r, where **r=1**, then C=Number of features in the dataset.
2. Minimum C: is obtained from maximum r, where **r**=$n/5$, then C=5.

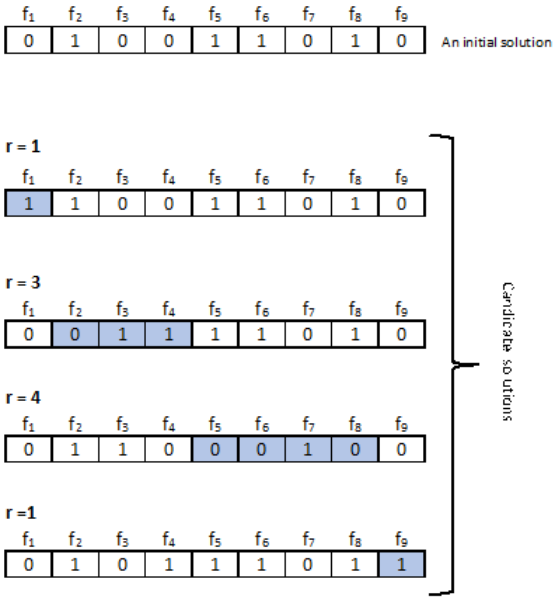

FIGURE 4.5: Random improvement operation

The deterministic improvement was not suitable for our work; since it doesn't enhance the solutions quality and not give adequate subset of selected features. Furthermore, this way takes long learning time and gives bad classification results. The random improvement gives high quality solutions and takes less time than the deterministic improvement.

- Reference set generation: After improvement the solutions, the reference set will be updated. This set ($RefSet$) has two sub sets which are $RefSet_1$ and $RefSet_2$,

the number of solutions in $RefSet_1$ set is greater than it in the $RefSet_2$ set [5]. The $RefSet_1$ set contains the best solutions from the initial population. On the other hand, the $RefSet_2$ set has the diverse solutions that have the maximum distances from the best solutions (i.e, the worst solutions in the initial population). The size of *RefSet* and $RefSet_1$ is predefined number then the size of $RefSet_2$ is $RefSetSize_2$=*RefSetSize*-$RefSetSize_1$.

- Solutions combination operation: this operation is like the crossover in the genetic algorithm. This operation is explained in the example in Figure 4.6. In this example r is a random a number that determines the point where to combined two solutions.

  Here, as shown in Figure 4.6, r=4, this means that the first 4 features of solution 1 are the first 4 features in the combined solution 1. Also, the first 4 features of solution 2 are the first 4 features in the combined solution 2. The rest features in solution 1 will be the rest features in the combined solution 2. Furthermore, the rest features in solution 2 will be the rest features in the combined solution 1.

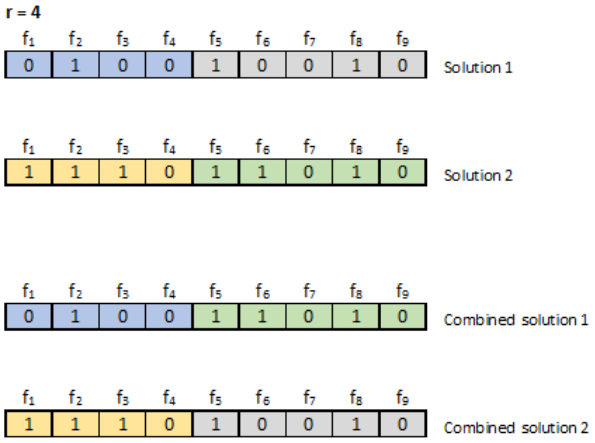

FIGURE 4.6: Solutions combination operation

In this thesis the solutions combination operation has been customized in a way that will enhance the solution quality. This operation is divided into three parts as the following:-

  - The first part: The best solutions are combined in this part, the number of solutions that results from this part is b_1.
  - The second part: The diverse (worst) solutions are combined in the second part, here there will be b_2 solutions.
  - The third part: In the last part the best and diverse solutions are combined, we combined each worst solution with one best solution. Here, the number of solutions is $2*b_2$.

- Remove redundant solutions: After combining the solutions and choosing the best solutions that result from the combination then, may be there will be redundant solutions. In our work we have implemented this operation that remove any redundant solutions in order to have a diverse solutions.

4.2.3 Cross validation

To evaluate the results of the proposed approach we split each dataset into two sets, learning and hidden sets. For the learning set we use 3-fold cross validation, The first fold is **70%** of samples for training, and the second fold is **30%** of samples for testing phase. In this thesis at every execution we randomly splitting the learning set between training and testing parts. The hidden set isn't used in training nor testing, it is used only for final evaluation of our proposed approach. For more information about the number of samples and the size for the learning data and hidden data in each dataset return to Table 4.1.

4.2.4 Performance evaluation

To measure the performance of the proposed approach; the classification results for the hidden data is compared with results with previous works which are used the same datasets, but these works used Accuracy (ACC) as a fitness function.

Area Under Curve (AUC) is better than ACC as proved in [45]; so in this work we used AUC as a fitness function to measure and evaluate the classification performance.

The formula for calculating the Accuracy as the following [8, 45]:-

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Where,

- TP (True Positive): is the number of positive samples that are classified correctly.
- FP (False Positive): is the number of negative samples that are classified as positive incorrectly.
- TN (True Negative): is the number of negative samples that are classified correctly.
- FN (False Negative): is the number of positive samples that are classified as negative incorrectly.

TABLE 4.2: Confusion matrix

	Positive samples (+)	**Negative samples (-)**
Yes	TP	FP
No	FN	TN

Table 4.2 is the confusion matrix [45] that illustrates the meaning of *TP*, *FP*, *TN*, and *FN*.

The formula for calculating the AUC as the following [45]:-

$$\mathbf{AUC} = \frac{\sum R(+) - (TP + FN)(FP + TN + 1)/2}{TP + TN + FP + FN} \tag{4.2}$$

Where, $\sum R(+)$ is the sum the R (Ranks) of all positively classified examples.

4.3 Sequential version of the proposed approach

As mentioned in section 4.2, the first step of the wrapper feature selection model, is using a search strategy in order to generate a subsets of selected features. In this research, we used the scatter search algorithm for this purpose. This section will discuss the sequential version of the scatter search.

In Figure 4.7 the basic operations of the sequential scatter search are illustrated. These operations are: generation initial population of solutions, improvement operation, evaluation, generation reference set of solutions and combined the solutions in the reference set. These steps are repeating until condition is met. The definition and the implementation of the SS algorithm's steps were discussed in details in section 4.2.

4.3.1 Theoretical time analysis

Here, we will explain the complexity of the sequential version of the scatter search algorithm that is used in this research. The Big-Oh notation will be used to describe the performance complexity (worst-case scenario) of an algorithm. For example, if there is n number of instances, going once through all the instances has the complexity of $O(n)$ [8].

SS complexity depends on its operations implementation, number of features in the dataset, population size, number of generations, and mainly on the fitness function.

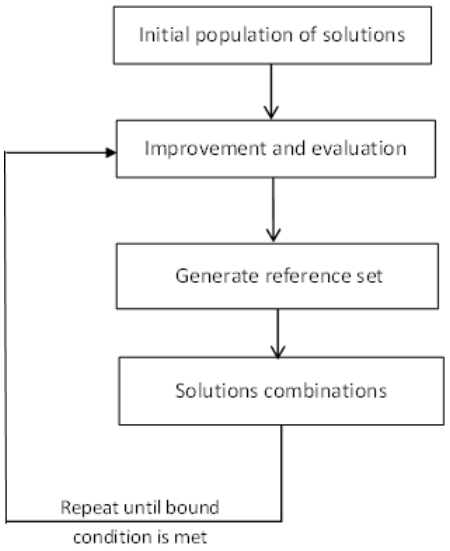

FIGURE 4.7: Sequential version of the scatter search algorithm

So, any change in these factors will affect on the overall complexity. In general, the sequential time complexity of the basic SS operations can be expressed as follow:-

$$O(T * (S * N) + S + S + (S * N)) \tag{4.3}$$

Where,
T: number of iterations.
S: number of solutions.
N: number of features.

The previous equation can be simplified to be:

$$O(T * (2 * S + 2 * S * N)) \tag{4.4}$$

Which is equal to:

$$O(2 * T * S + 2 * T * S * N) \tag{4.5}$$

The complixity of sequential SS algorithm is:

$$O((2TSN)(N + 1)) \tag{4.6}$$

4.4 Parallel version of the proposed approach

Section 4.3 discussed the sequential version of scatter search algorithm. The second version of the proposed approach is the parallel scatter search algorithm, which will be discussed in this section. Then the communication and synchronization between master and worker processes is explained. Finally, we will talk about the time complexity for the parallel version of scatter search that is used in this research.

4.4.1 The parallel model for SS algorithm

In the SS algorithm the improvement operation takes the longest execution time comparing with other operations in the SS algorithm. Based on this reason when the parallel version of SS was implemented, the improvement operation was distributed between the worker processes. On the other hand the rest operations of SS algorithm was executed by the master process.

The master-worker parallel architecture that we used in this research is illustrated in Figure 4.8. The master process initializes solutions and distributes solutions, then while the workers do their tasks the master applies improvement and evaluation on it's solutions. Each Worker process applies improvement operation, evaluates the solutions, then returns the best solution to the master process. When the workers finished their tasks then the master process collects solutions from workers, updates Reference Set, combines solutions. After that the combined solutions are distributed between workers for improvement and evaluation.

4.4.2 Communication and synchronization

Communication and coordination (synchronization) are crucial factors to collaborate cores in an efficient manner so that errors do not occur. In the following we demonstrate the aspects of communication and synchronization that were used in our study.

Communication: We used point-to-point communication; which is communication between two processes. To complete this task the following steps are needed:-

1. Solutions distribution:
    - Before this operation starts, the master process divides the solutions equally between the worker processes, then it gets the reminder solutions.
    - The distributions of solutions among the worker processes, there are two messages must be sent to each worker:-

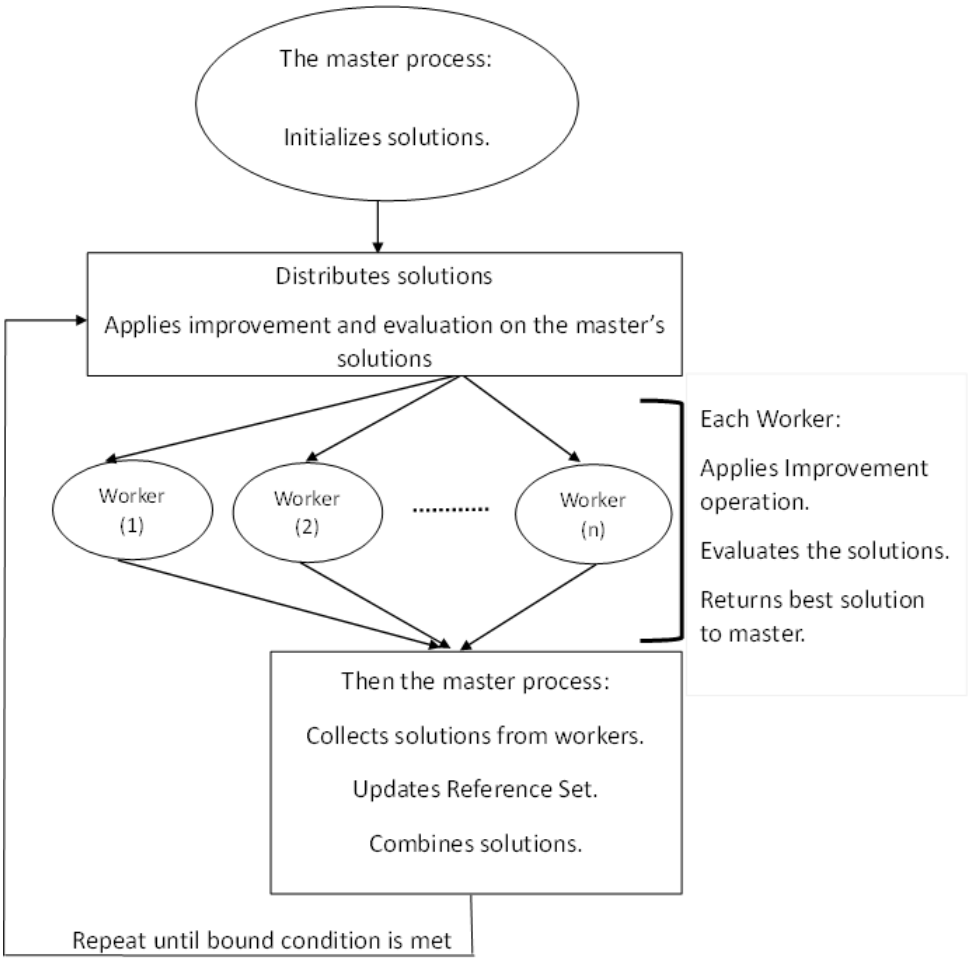

FIGURE 4.8: Parallel version of the scatter search algorithm

- The first message includes the number of solutions that will be sent to the worker.
- The master will send the second message which includes the solutions that will be improved at this worker.

2. While the workers processes doing their jobs, the master process applies improvement operation on its solutions.

3. Solutions collection:

   - When the worker process applied the improvement operation on its solutions, it will send a message to the master, this message includes the best solution that results from applying the improvement operation on each solution of the worker's solutions.

Number of needed messages: Suppose we have number of processors is P, and S solutions. On average there will be S/P solutions per processor in every iteration of the

algorithm. Each of the *(P-1)* worker process will need $\left(\frac{2S}{P}+2\right)$ messages (for each iteration) to receive its *S/P* solutions and return back the improved solutions to the master process.

Synchronization: There is a synchronization point among processes. So, all processes must reach a point in their code before they can all begin executing again. This point when all workers processes finished their job and send a flag message to the master. When the master receives this message from all workers, then the next iteration will start.

4.4.3 Theoretical time analysis

As mentioned previously, the time complexity for the sequential SS algorithm is: $O(2TSN)$. Here the time complexity for the parallel SS algorithm is:-

$$\Delta + O\left(\frac{2TSN}{P}\right) \tag{4.7}$$

Where Δ is the overhead of the used parallel hierarchy, and P is the number of the processors. The overhead in the parallel SS algorithm comes from two sources, they are:-

- In each iteration there are two times of improvement operation, so there are two times of communication and synchronization between master and workers.
- The machine that was used in this work is a virtual server not physical.

Chapter 5

Experiments and results

This chapter presents the experimental results of the proposed approach, work environment and experimental settings. Furthermore, the algorithm enhancement, the results of the sequential and the parallel versions of the proposed approach will be discussed.

5.1 Experimental environment settings

In the following steps we describe the experiments setup:-

- **Parallel computing environment**
  Due to the fact that we are working on high dimensional datasets; our experiments were run on a virtual server with 32 processors, each has 2200.026 MHz CPU, 512 KB cache size and the total memory 24.6 GB.

  This server operates on *CentOSLinux*, with all needed programming libraries and tools for C and R. The parallel programming paradigm was used is the Open Message Passing Interface (Open MPI) V1.10.2 with all support libraries and tools.

- **Open MPI**
  Among the distributed memory programming models, the *MPI* model is commonly used to parallelize applications. *MPI* is an explicit programming model. The programmer implements the distribution of the tasks, communication between them, and decides how the work is allocated between the various threads [33].

  With the emergence of multi-core systems, hybrid programming models have also been developed. Within a single node, fast communication through shared memory can be exploited, and a networking protocol can be used to communicate across the nodes. Programs can then take advantage of both shared memory and distributed memory [33].

TABLE 5.1: Best classification results obtained from the deterministic improvement operation in the sequential SS algorithm using RF or SVM classifiers

Dataset	Arcene	Madelon
AUC in Training	0.73	0.56
AUC in Testing	0.70	0.50

5.2 Algorithm enhancement

This section discusses the enhancement of the algorithm, in particular how we implemented and customized the improvement operation in order to enhance the classification performance for high dimensional datasets and also reduce the execution time. This section will compare the results of deterministic improvement and random improvement.

5.2.1 Deterministic improvement operation

Table 5.1 shows the results of AUC for two datasets which are Arecene and Madelon. The classification performance was **0.73** for Arcene dataset in training phase for training data, and **0.70** in testing phase for hidden data. For Madelon dataset the results were: **0.56** for train data in training phase, and **0.50** in testing phase for hidden data. Which means that bad results comparing with previous works in [8] and [1].

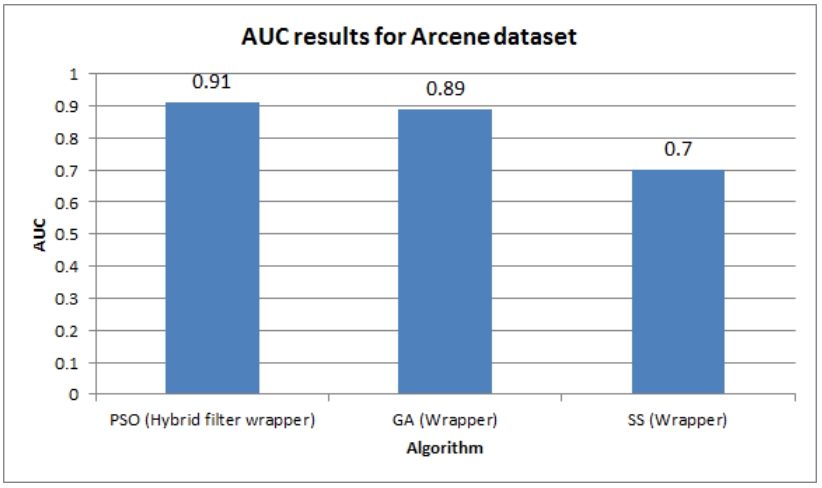

FIGURE 5.1: Comparing the classification results for Arcene dataset in the testing phase of deterministic improvement for SS with other algorithms

Figure 5.1 and Figure 5.2 illustrate the results of previous works [8], [1] and the results of AUC of the deterministic improvement method that was used in this research. Figure 5.1, shows the AUC results for Arcene datasets. The PSO algorithm with hybrid wrapper-filter approach has the best result **0.91**. While the SS has the worst results **0.70**. From Figure 5.2, it is noticeable that the genetic algorithm with wrapper approach

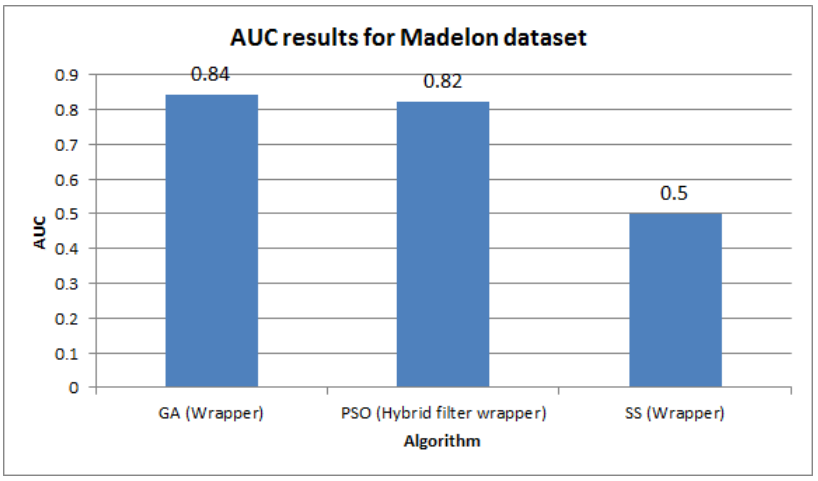

FIGURE 5.2: Comparing the classification results for Madelon dataset in the testing phase of deterministic improvement for SS with other algorithms

TABLE 5.2: Best classification results obtained from the random improvement operation in the sequential SS algorithm

Dataset	**Arcene (SVM classifier)**	**Madelon (RF classifier)**
AUC in Training	0.99	0.92
AUC in Testing	0.93	0.92

that was used in [8] has the best AUC result **0.84** for Madelon dataset; while the SS has the worst results **0.50**.

5.2.2 Random improvement operation

After we customized the improvement operation, the AUC results become better, these results are listed in Table 5.2. The classification performance was **0.99** for Arcene dataset in training phase for training data, and **0.93** in testing phase for hidden data. For Madelon dataset the results were: **0.94** for train data in training phase, and **0.92** in testing phase for hidden data. Which means that best results comparing with previous works in [8] and [1].

Figure 5.3 and Figure 5.4 illustrate the results of previous works [8], [1] and the results of AUC of the random improvement method that was used in this research. Figure 5.3, shows the AUC results for Arcene datasets. The SS algorithm with wrapper approach has the best result **0.93**. Also from Figure 5.4, it is noticeable that the SS algorithm with wrapper approach has the best AUC result **0.92** for Madelon dataset.

The above results and discussion explain how the customized improvement operation enhanced the classification performance. Furthermore, since the random improvement operation produce random number of candidate solutions; this means that we don't

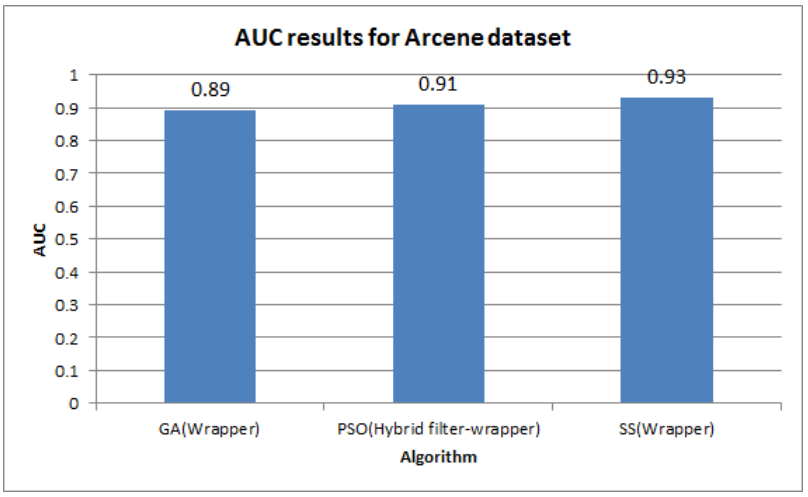

FIGURE 5.3: Comparing the classification results for Arcene dataset in the testing phase of random improvement for SS with other algorithms

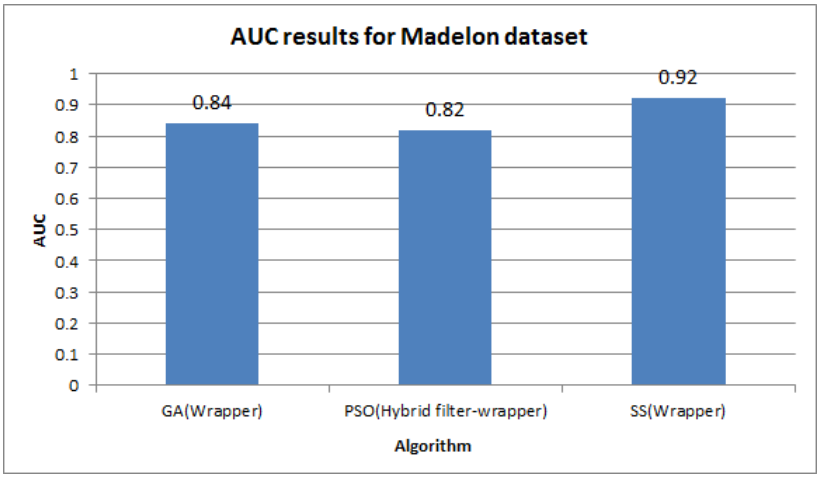

FIGURE 5.4: Comparing the classification results for Madelon dataset in the testing phase of random improvement for SS with other algorithms

TABLE 5.3: Comparing the learning time for the deterministic and random improvement operation

Dataset	**Arcene**	**Madelon**
Learning time for deterministic improvement	30 hours	48 hours
Learning time for random improvement	14 hours	22 hours

evaluate all possible candidate solutions. The needed learning time will be less than the time that is needed when the deterministic improvement is used, see Table 5.3.

5.3 Sequential scatter search for feature selection in high dimensional datasets

This section will present and discuss the results of classification performance (AUC and execution time) for the sequential SS algorithm for feature selection in high dimensional datasets. Furthermore, we will compare our results of sequential SS with other works that used the same datasets.

5.3.1 Arcene dataset's results and discussion

This section will discuss the classification results for Arcene datasets, and comparing the classification performance of the sequential SS that built on wrapper approach with the GA built on wrapper approach [8], and with PSO algorithm that built on hybrid filter wrapper approach [1].

Figure 5.5 presents the classification results for Arcene dataset using *Random Forest (RF)* classifier. Here we compare the sequential version of SS that used in this research with GA that built on wrapper approach [8] and with research in [1] that used PSO and built using filter wrapper approach. The results were: the SS and the GA has **0.85**, while the PSO has **0.89**.

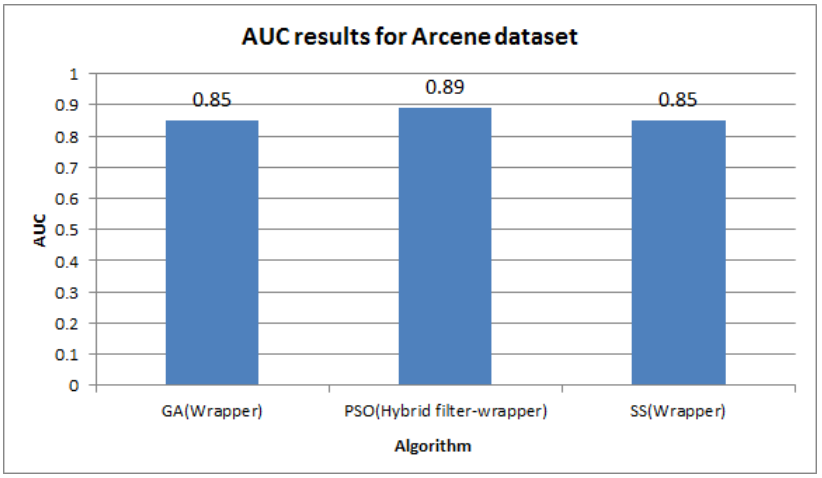

FIGURE 5.5: Classification results for Arcene dataset using RF classifier with different algorithms in the testing phase

Figure 5.3 in section 5.2.2, presents the classification results for Arcene dataset using *Support Vector Machine (SVM)* classifier. Here we also compare the sequential version of SS that used in this research with GA that built on wrapper approach [8] and with research in [1] that used PSO and built using filter wrapper approach. The SS has **0.93**,

TABLE 5.4: Comparing the classification results for Arcene dataset in the testing phase

Algorithm	RF classifier	SVM classifier
GA (Wrapper)	0.85	0.89
PSO (Hybrid filter)	0.89	0.91
SS(Wrapper)	0.85	0.93

the PSO has **0.91**, while GA has **0.89**. We can notice that the proposed approach that used SS has enhance the classification performance for Arcene dataset.

Table 5.4 compares the SS with previous works in [8] and [1] for Arcene dataset when using RF and SVM classifiers in evaluation phase. Our approach has the best classification result (AUC = **0.93**) for Arece datasets when we used SVM classifier.

5.3.2 Madelon dataset's results and discussion

This section will discuss the classification results for Madelon datasets, and comparing the classification performance of the sequential SS that built on wrapper approach with the GA built on wrapper approach [8], and with PSO algorithm that built on hybrid filter wrapper approach [1].

Figure 5.4 in section 5.2.2, presents the classification results for Madelon dataset using Random Forest (RF) classifier. Here we compare the sequential version of SS that used in this research with GA that built on wrapper approach [8] and with research in [1] that used PSO and built using filter wrapper approach. The results were:the SS has **0.92**, the GA has **0.84**, while PSO has **0.82**. It is noticeable that the proposed approach that used SS has enhance the classification performance for Madelon dataset.

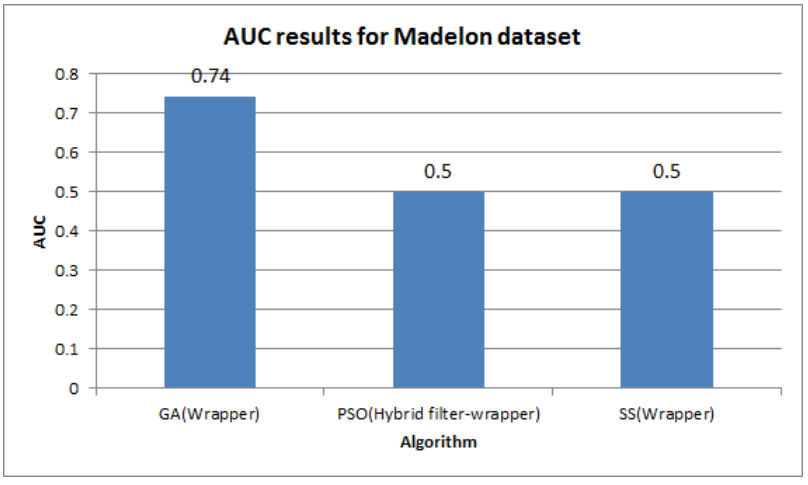

FIGURE 5.6: Classification results for Madelon dataset using SVM classifier with different algorithms in the testing phase

TABLE 5.5: Comparing the classification results for Madelon dataset in the testing phase

Algorithm	RF classifier	SVM classifier
GA (Wrapper)	0.84	0.74
PSO (Hybrid filter)	0.82	0.50
SS(Wrapper)	0.92	0.50

Figure 5.6 presents the classification results for Madelon dataset using Support Vector Machine (SVM) classifier. Here we also compare the sequential version of SS that used in this research with GA that built on wrapper approach [8] and with research in [1] that uses PSO and built using filter wrapper approach. The SS has **0.50**, the PSO has **0.50**, while GA has **0.74**.

Table 5.5 compares the SS with previous works in [8] and [1] for Madelon dataset when using RF and SVM classifiers in evaluation phase. Our approach achieves the best classification result (AUC = **0.92**) for Madelon datasets when the RF classifier is used.

5.3.3 Mushroom dataset's results and discussion

Mushroom dataset is the smallest dataset, the classification result was **1.0** for the approaches: SS, GA, and PSO. Furthermore this is the same result for RF or SVM classifier. In Figure 5.7 the AUC is **1.0** for the three algorithms.

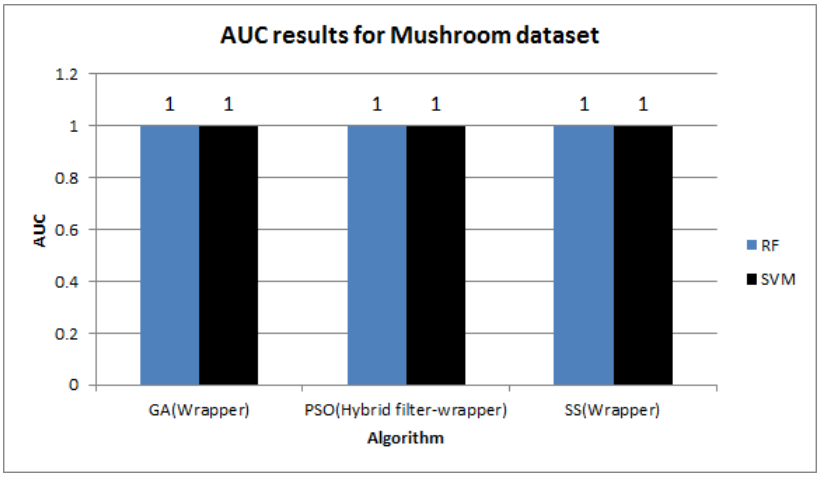

FIGURE 5.7: Classification results for Mushroom dataset for different classifiers (RF, SVM) and algorithms in the testing phase

5.3.4 Learning time

This section will present and compare the learning time for SS, and GA algorithms for Arcene, and Madelon datasets. We will compare SS and GA only; since the both

algorithms are built on the wrapper approach.

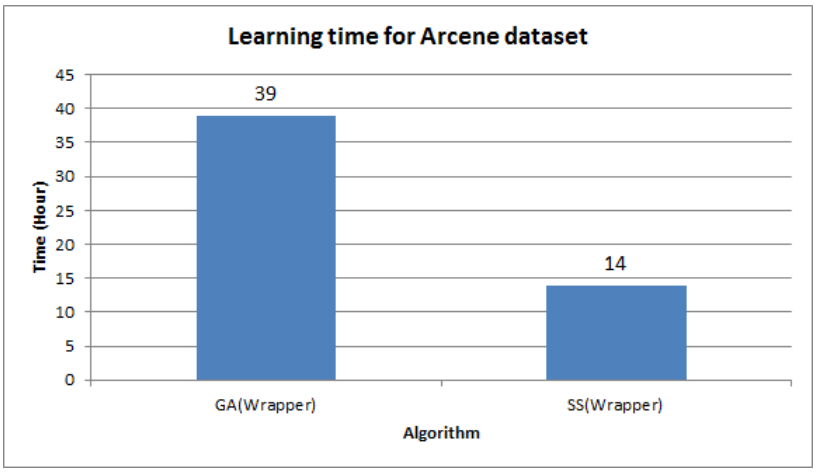

FIGURE 5.8: Comparing learning time (Hour) for Arcene dataset for GA and SS algorithms

Figure 5.8 depicts the learning time (Hour) for Arcene dataset. In GA the learning time is **39** hours, while in the SS is reduced to **14** hours. This result shows the efficiency of SS algorithm in the learning time for Arcene dataset.

Figure 5.9 illustrates the learning time (Hour) for madelon dataset. In GA the learning time is **74** hours, while in the SS is reduced to **22** hours. Also, this result shows the efficiency of SS algorithm in the learning time for Madelon dataset.

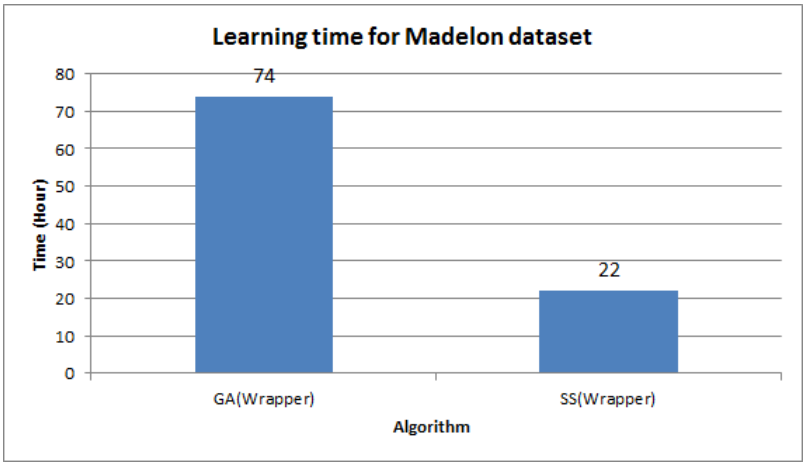

FIGURE 5.9: Comparing learning time (Hour) for Madelon dataset for GA and SS algorithms

5.4 Parallel scatter search for feature selection in high dimensional datasets

In section 5.3 we show the efficiency of sequential SS comparing with others works in term of execution time and classification performance. This section will present and discuss the results of classification performance (AUC and execution time) for the parallel SS algorithm for feature selection in high dimensional datasets. Finally, we will compare the learning time for the sequential version versus the the parallel version of the proposed approach.

5.4.1 Arcene dataset

The classification performance for Arcene dataset is illustrated in Figure 5.10. The AUC in learning phase is increased from **0.99** in sequential SS to **1.0** in parallel SS. On the other hand, the AUC in testing phase is increased from **0.93** in sequential SS to **0.94** in parallel SS.

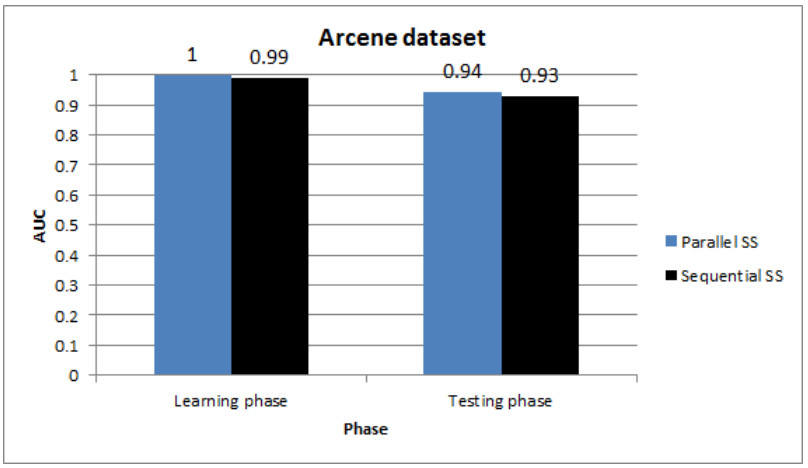

FIGURE 5.10: Comparing classification performance for Arcene dataset

Figure 5.11 compares the learning time for Arcene dataset in sequential SS and parallel SS. It is clear that the parallel version enhanced the performance for the learning time. The learning time for Arcene dataset in sequential SS was **14** hours; but this time is reduced to **2** hours in parallel SS.

5.4.2 Madelon dataset

This part will compare and discuss the results for Madelon dataset in sequential SS and parallel SS. The classification performance for Madelon dataset is illustrated in Figure

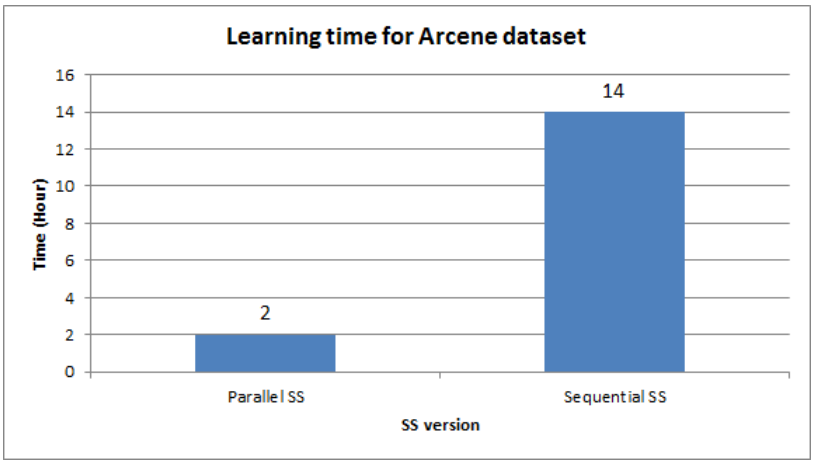

FIGURE 5.11: Comparing learning time (Hour) for Arcene dataset for sequential and parallel SS algorithm

5.12. The AUC has the same values in learning and testing for sequential SS and parallel SS.

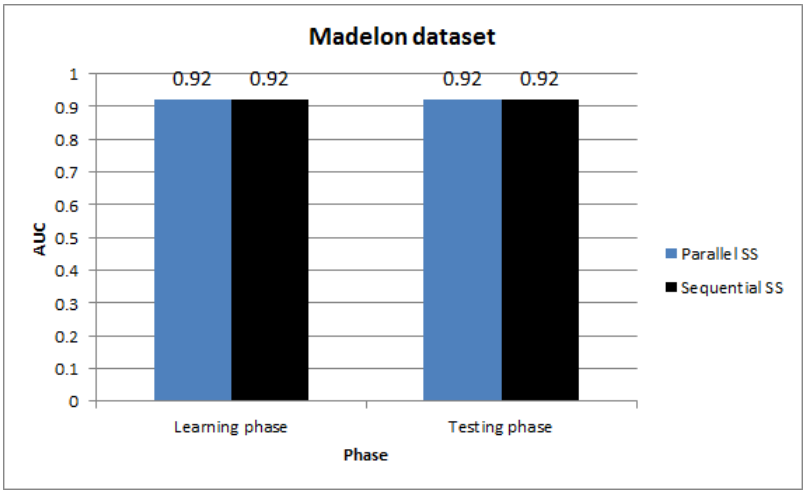

FIGURE 5.12: Comparing classification performance (AUC)for Madelon dataset

Figure 5.13 compares the learning time for Madelon dataset in sequential SS and parallel SS. It is clear that the parallel version enhanced the performance for the learning time. The learning time for Madelon dataset in sequential SS was **22** hours; but this time is reduced to **3** hours in parallel SS.

5.5 Spambase dataset

This part will compare the classification's results for the Spambase dataset with the results in work [43]. In this research they used Artificial neural networks (HC-RBFN-PSO). Figure 5.14 shows that the proposed approach has better classification performance for

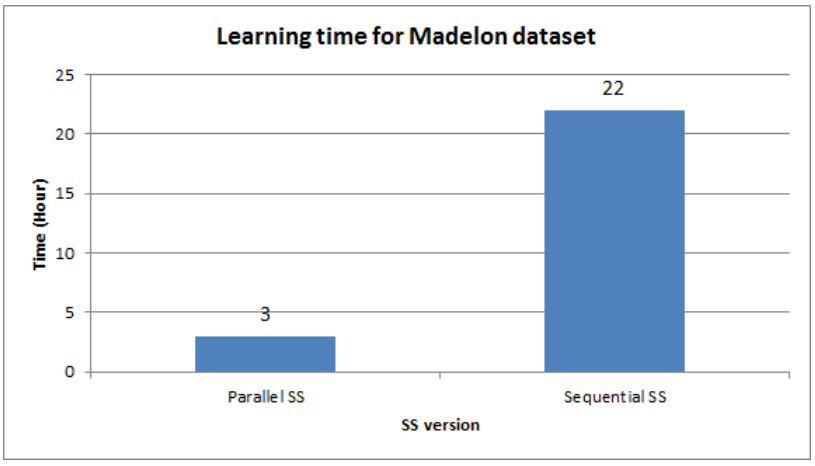

FIGURE 5.13: Comparing learning time (Hour) for Madelon dataset for sequential and parallel SS algorithm

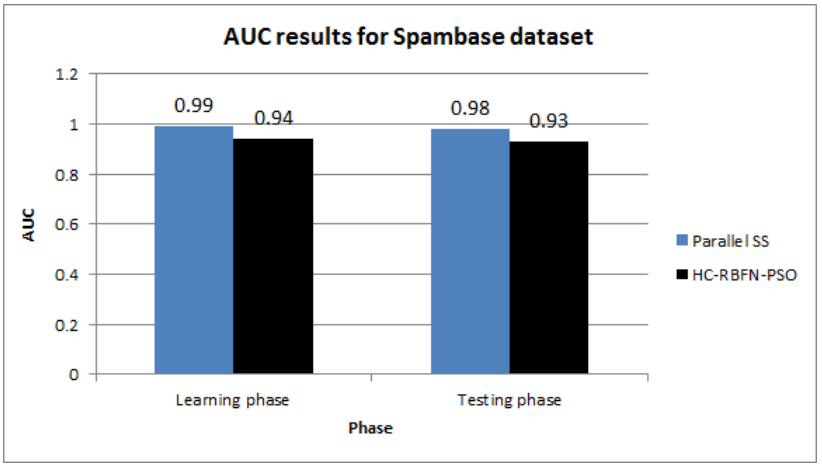

FIGURE 5.14: Comparing classification performance for Spambase dataset for different algorithms

Spambase dataset in learning and testing phases. For Spambase dataset, in learning phase our approach has (AUC = **0.99**), while the HC-RBFN-PSO has **0.94**. Furthermore, in testing phase the classification result was **0.93** for HC-RBFN-PSO, and **0.98** for our approach.

5.6 Gisette dataset

This section compares the classification's results for the Gisette dataset with the results in work [44]. In this research they are used parallel implementation in R of the weighted subspace random forest algorithm. Figure 5.15 presents that the proposed approach has better classification performance for Gisette dataset. Our approach has (AUC = **0.99**), while the parallel RF has (AUC = **0.97**).

Figure 5.16 compares the learning time for Gisette dataset in sequential SS and parallel SS. It is clear that the parallel version enhanced the performance for the learning time. The learning time for Gisette dataset in sequential SS was **120** hours; but this time is reduced to **40** hours in parallel SS.

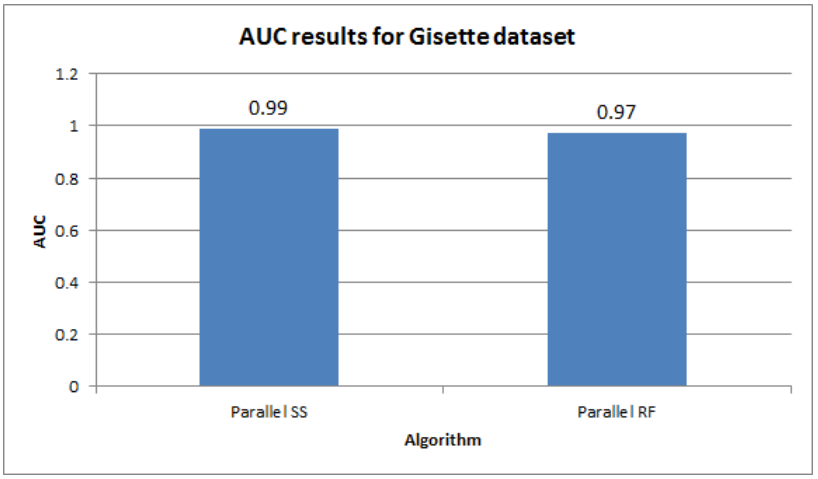

FIGURE 5.15: Comparing classification performance for Gisette dataset for different algorithms in the testing phase

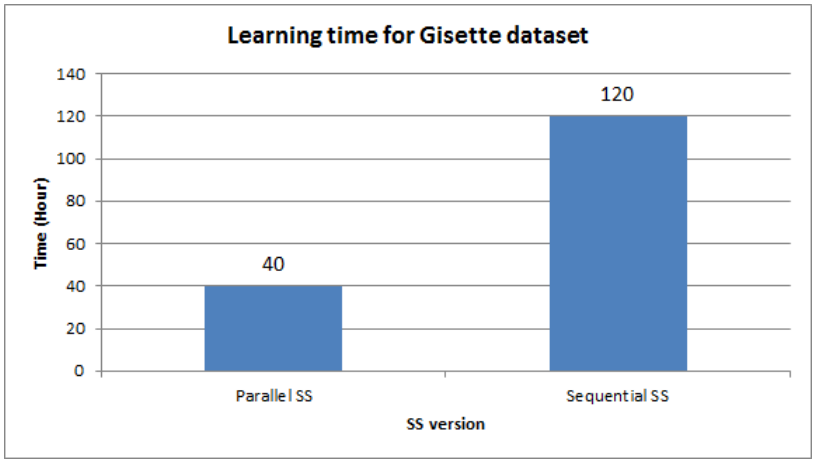

FIGURE 5.16: Comparing learning time (Hour) for Gisette dataset for sequential and parallel SS algorithm

5.7 Time analysis and performance

While we are using the parallel hierarchy, the number of cores we operate affects the convergence time. The server that we used to run our experiments has **32** cores. According to our parallel methodology, one core is the master core and the other cores are the workers.

In the following subsections, we will handle the relationship between the parallel execution time and the number of running parallel cores, then we will discuss the speedup performance.

5.7.1 Convergence of parallel execution time

Figure 5.17 shows the relationship between the number of running parallel cores and the average convergence time. We found that as much as we increase the number of running parallel cores, the algorithm converges faster. The performance does not have a significant increment from **16** threads to **32** cores due to communication overhead and virtual resource limitation especially memory size.

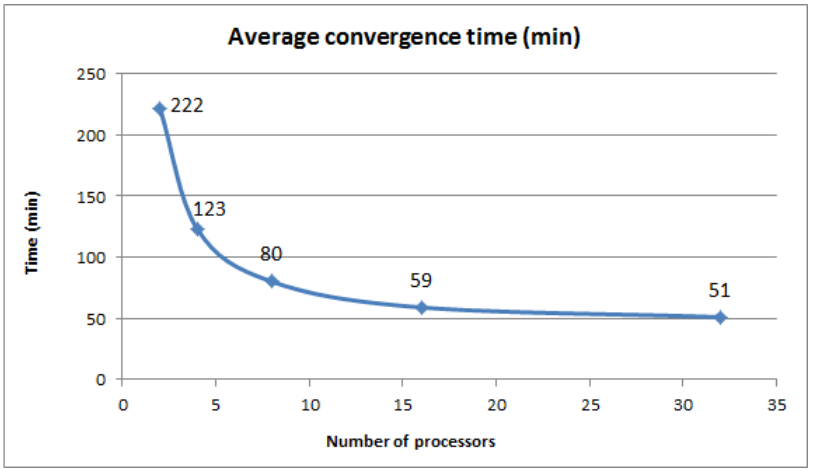

FIGURE 5.17: The parallel execution time with different number of parallel cores

5.7.2 Empirical speedup

In computer architecture, speedup is the improvement in speed of execution of a task executed on two similar architectures with different resources [46, 47]. In this section we will discuss the speedup performance for the datasets that we used in our experiments.

The sequential execution time, parallel execution time and the performance speedup for five datasets that used in the experiments are listed in Table 5.6.
To compute the speedup we use Equation 5.1 [48].

$$\textbf{Speedup} = \frac{SET}{PET} \tag{5.1}$$

Where:
SET: Sequential Execution Time.
PET: Parallel Execution Time.

TABLE 5.6: Summary of: sequential execution time, parallel execution time and performance speedup for different datasets

Dataset	Sequential execution time (hour)	Parallel execution time (hour)	Speedup
Mushroom	1	0.5	2x
Spambase	1.5	0.67	2.2x
Arcene	14	2	7x
Madelon	22	3	7.3x
Gisette	120	40	3x

Figure 5.18 shows the performance speedup of using the proposed approach when it is running with various types of datasets. For Mushroom dataset the PET less than the SET by **2x**. For Spambase dataset the PET less than the SET by **2.2x**. For Gisette dataset the PET less than the SET by **3x**. For Arcene dataset the PET less than the SET by **7x**. Finally, for Madelon dataset the PET less than the SET by **7.3x**.

The speedup is increased as the complexity of the dataset is increased. We noted that the effect of overhead in the virtual machine which comes from limited memory size and communication latency, all that limits the maximum speedups gained.

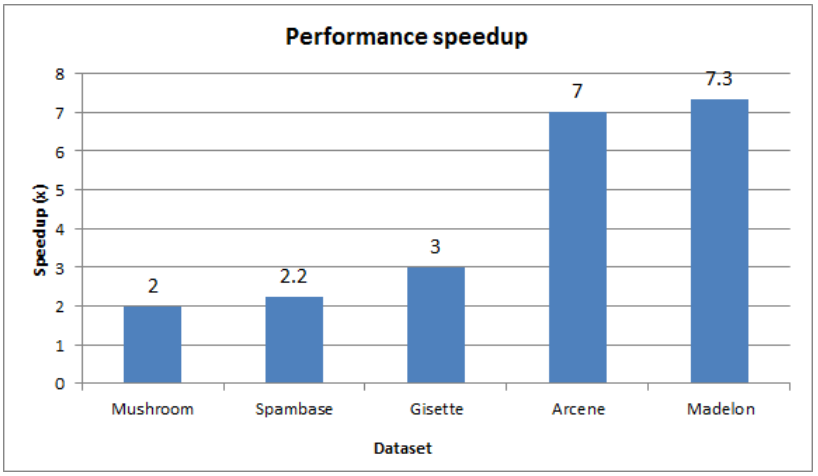

FIGURE 5.18: Performance speedup of parallel SS with different datasets

Chapter 6

Conclusion and future work

In this chapter we will conclude this thesis, and discuss the future work perspectives.

6.1 Conclusion

In this thesis we developed two versions of the scatter search algorithm (sequential and parallel) using wrapper model. This approach enhance the classification performance for high dimensional datasets.

The results show the efficiency of SS algorithm comparing with other evolutionary algorithms that are deal with feature selection in high dimensional datasets. This approach was tested on five datasets: Mushroom, Spambase, Arcene, Madelon and Gisette. Three of them (Arcene, Madelon and Gisette) are feature selection challenge.

The main results of the experiments can be summarized in these points:-

- For Mushroom datasets the classification results of our experiments are equivalent to the results of previous works that used GA with wrapper model and PSO that built using Hybrid filter wrapper model, it is **1.0** for RF and SVM classifiers, the number of selected features in the best solutions is **13**.
- For Spambase dataset our result was better than the results of the work that used Neural Networks with PSO algorithm, we enhanced the AUC results from **0.93** to **0.98** for RF and SVM classifiers, the number of selected features in the best solutions is **26**.
- For the other three datasets which are feature selection challenge, the results are as follow:-

  - For Arcene dataset, we enhanced the AUC result comparing with works that used GA and PSO algorithms. We increased the AUC results to **0.93** for SVM classifier, the number of selected features in the best solutions is **4971**. While in GA the result was **0.89**, and the AUC in PSO was **0.91**. Also, for Arcene dataset we reduced the execution time comparing with the works that used GA with wrapper model we reduced the learning time from **39** hours to **14** hours in the sequential SS algorithm.

  - For Madelon dataset we enhanced the AUC result comparing with works that used GA and PSO algorithms. We increased the AUC results to **0.92** for RF classifier, the number of selected features in the best solutions is **256**. While in GA the result was **0.84**, and the AUC in PSO was **0.82**. Also, for Madelon dataset we reduced the execution time comparing with the works that used GA with wrapper model we reduced the learning time from **74** hours to **22** hours in the sequential SS algorithm.

  - The last feature selection challenge dataset that we used in our experiments is Gisette. For this dataset we enhanced the classification performance results from **0.97** to **0.99** for RF and SVM classifiers, the number of selected features in the best solutions is **2510**, comparing with previous work that used R Package for classification with scalable weighted subspace Random Forests.

- The AUC results of parallel and sequential versions of the SS algorithm are similar for Mushroom, Spambase, Madelon and Gisette datasets. For Arcene dataset the parallel version of SS increased the AUC from **0.93** to **0.94**.

- The parallel version of SS algorithm reduced the execution time for the five datasets. For Mushroom dataset the execution time was reduced from **1.0** hour to **0.5** hour. For Spambase dataset the execution time was reduced from **1.5** hours to **0.67** hour. For Arcene dataset the execution time was reduced from **14** hours to **2** hours. For Madelon dataset the execution time was reduced from **22** hours to **3** hours. Finally, for Gisette dataset the execution time was reduced from **120** hours to **40** hours.

6.2 Future work

In this thesis we used SS algorithm with wrapper model to handle the classification problem in high dimensional datasets. In future we will:-

- Use the number of selected features in fitness function.
- Apply the hybrid filter wrapper model with SS algorithm and comparing the results with wrapper model.
- Use SS algorithm in other applications such as prediction.

Bibliography

[1] Fatima Koumi, Mohammed Aldasht, Hashem Tamimi. "Efficient feature selection using binary particle swarm optimization A hybrid filter approach". *Master Thesis, Palestine Polytechnic University*, 2016.

[2] E. Albaa, G. Luquea, S. Nesmachnowb. "Parallel metaheuristics: recent advances and new trends". *Intl. Trans. in Op. Res, DOI: 10.1111/j.1475-3995.2012.00862.x, Volume 20, Pages 1-48*, 2013.

[3] O. Abedinia, N. Amjady, H. Zareipour. "A New Feature Selection Technique for Load and Price Forecast of Electrical Power Systems". *IEEE, DOI 10.1109/TPWRS.2016.2556620, Volume 32, NO. 7, Pages 62-74*, 2017.

[4] J. Garca-Nieto, E. Alba. "Parallel multi-swarm optimizer for gene selection in DNA microarrays". *DOI 10.1007/s10489-011-0325-9, Appl Intell, Volume 37, Pages 255-266*, 2012.

[5] F. Lopez, M. Torres, B. Batista, J. Perez, J. Vega. "Solving feature selection problem by a parallel Scatter Search". *Elsevier, European Journal of Operational Research, Volume 169, Pages 477-489*, 2006.

[6] Y. Zhu, J. Liang, J. Chen, Z. Ming. "An improved NSGA-III algorithm for feature selection used in intrusion detection". *Elsevier, Volume 116, Pages 74-85*, 2017.

[7] A. Mohamed. "An efficient modified differential evolution algorithm for solving constrained non-linear integer and mixed-integer global optimization problems". *International Journal of Machine Learning and Cybernetics, Volume 8, NO. 3, Pages 898-1007*, 2017.

[8] Murad Alkubaji, Mohammed Aldasht. "One class genetic-based feature selection for classification in large datasets". *Master Thesis, Palestine Polytechnic University*, 2015.

[9] J. Benitez, F. Herrera D. Peralta, S. RA.O, S. Rama.rez-Gallego, I.Triguero. "Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach".

Hindawi Publishing Corporation, Mathematical Problems in Engineering, Volume 11, Pages 1-11, 2015.

[10] J. Qian, D. Miao, Z. Zhang, X. Yue. "Parallel attribute reduction algorithms using MapReduce". *Elsevier, Information Sciences, Volume 279, Pages 671-690*, 2014.

[11] J. Liu, H. Iba. "Selecting Informative Genes with Parallel Genetic Algorithms in Tissue Classification". *Genome Informatics, Volume 12, Pages 14-23*, 2001.

[12] http://whatis.techtarget.com/definition/machine learning. (2017, March 25).

[13] M. Topalovic, J. Aerts, M. Decramer, T. Troosters, W. Janssens. "Artificial Intelligence Detects Lung Diseases Using Pulmonary Function Tests". *American Journal of Respiratory and Critical Care Medicine, ATS Conferences ATS 2017 C47. COPD: PHYSIOLOGIC ASSESSMENT*, 2017.

[14] A. Cano, J. Olmo, S. Ventura. "Parallel Multi-Objective Ant Programming for Classification Using GPUs". *Journal of Parallel and Distributed Computing*, Volume 73, NO. 6, June 2013, Pages 713-728.

[15] Z. Zhang, P. yang. "An Ensemble of Classifier with Genetic Algorithm based Feature Selection". *IEEE Intelligent Information Bulletin, Volume 9, NO. 1, Pages 18-24*, pages 18–24, 2008.

[16] C. Guerra salcedo, S. Chen, D. Whitley, S. Smith. "Fast and Accurate Feature Selection Using Hybrid genetic Strategies". *IEEE Xplore*, Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on.

[17] L. Breiman. "Random Forests". *Springer, ISSN 1573-0565, Volume 45, NO. 1, Pages 532*, 2001.

[18] H. Liu, M. Hiroshi. "Instance Selection and Construction for Data mining". *Springer ,Eds, ISSN 978-1-4757-3359-4, Pages 3-20*, 2001.

[19] A. Onan, S. Korukog. "A feature selection model based on genetic rank aggregation for text sentiment classification". *Journal of Information Science, DOI: 10.1177/0165551515613226, Volume 43, NO. 4, Pages 1-14*, 2015.

[20] H. Ling Chen, B. Yang, S. Wang, G. Wang, D. Liu, H. Zhong Li, W. Liu. "Towards an optimal support vector machine classifier using a parallel particle swarm optimization startegy". *Applied Mathematics and Computation. Elsevier, Volume 239, Pages 180-197*, 2014.

[21] A. Das, S. Das. "Feature weighting and selection with a Pareto-optimal trade-off between relevancy and redundancy". *Elsevier, Volume 88, Pages 12-19*, 2017.

[22] M. Adamczyk. "Parallel Feature Selection Algorithm based on Rough Sets and Particle Swarm Optimization". *DOI:.10.15439/2014F389, ACSIS, IEEE Xplore, Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, 2014.

[23] X. Zhou, X. Gao, J. Wang, H. Yu, Z. Wang, Z. Chi. "Eye tracking data guided feature selection for image classification". *Elsevier, Volume 63, NO. 16, Pages 56-70*, 2017.

[24] O. Soufan, D. Kleftogiannis, P. Kalnis, V. Bajic. "DWFS: A wrapper Feature Selection Tool Based on a Parallel Genetic Algorithm". *PLOS ONE, DOI: 10.1371/journal.pone.0117988, Pages 1-23*, 2015.

[25] F. Glover. "Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms)". *Elsevier, Discrete Applied Mathematics, Volume 49, NO. 3, Pages 231-255*, 1994.

[26] F. Glover. "A Template For Scatter Search And Path Relinking". *Springer, Artificial Evolution, Volume 1363, Pages 1-51*, 1998.

[27] F. Glover, M. Laguna, R. Mart. "Scatter Search, in Advances in Evolutionary Computation: Theory and Applications". *Springer-Verlag, Pages 519-537*, 2003.

[28] F. Glover, M. Laguna, R. Mart. "Fundamentals of Scatter Search and Path Relinking". *Control and Cybernetics, ISSN 0324-8569 Vol. 29, NO. 3, Pages 653-684*, 2000.

[29] M. Laguna, R. Mart. "Scatter search: methodology and implementations in C". *"Springer Science Business Media, Volume 24, ISSN 146150337X, Pages 291-300*, 2012.

[30] R. Mart, M. Laguna, F. Glover. "Principles of Scatter Search". *Elsevier, European Journal of Operational Research, Volume 169, NO. 2, Pages 359-372*, 2006.

[31] Azeeza Salameh, Hashem Tamimi, Mohammed Aldasht. "Optimization of High Dimensional Data Visualization Using Parallel Genetic Algorithms". *Master Thesis, Palestine Polytechnic University*, 2016.

[32] A. Gottlieb, G.S. Almasi. "Highly Parallel Computing". *"Benjamin-Cummings Publishing Co., Inc. Red word city, CA, USA, ISSN 5217482*, 1989.

[33] M. AbuTaha, S. El Assad, A. Queudet, O. Deforge. "Design and Efficient Implementation of a Chaos-based Stream Cipher". *International Journal of Internet Technology and Secured Transactions, Paper IJITST*$_1$61464, *ResearchGate*, 2017.

[34] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. "A high-performance, portable implementation of the mpi message passing interface standard". *Parallel computing, Volume 22, NO. 6, Pages 789-828*, pages 789–828, 1996.

[35] J. Petrlik, L. Sekanina. "Towards Robust and Accurate Traffic Prediction Using Parallel Multiobjective Genetic Algorithms and Support Vector Regression". *IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.

[36] B. Hoai Nguye, B. Xue, P. Andreae. "A Novel Binary Particle Swarm Optimization Algorithm and Its Applications on Knapsack and Feature Selection Problems". *Springer, Intelligent and Evolutionary Systems, ISSN 978-3-319-49049-6, Pages 319-332*, 2017.

[37] A. Sameh, A. Ayman, N. Hasan. "Parallel Ant Colony Optimization". *International Journal of research and reviews in Computer Science (IJRRCS), Volume 1, NO. 2, Pages 77-82*, 2010.

[38] L. Zheng, Y. Lu, M. Ding, Y.Shen, M. Guo, S. Guo. "Architecture-based performance evaluation of genetic algorithms on multi/many core systems". *The 14th IEEE International Conference on Computational Science and Engineering. 978-0-7695-4477-9/11, (2011) IEEE DOI 10.1109/CSE.2011.65*, 2011.

[39] J. Pinho, L. Sobral, M. Rocha. "Parallel evolutionary computation in bioinformatics applications". *Elsevier. Computer methods and programs in biomedicine, Volume 110, Pages 183-191*, 2013.

[40] M. Meena, K.R. Chandran, A. Kathik, A. Samuel. "A parallel ACO algorithm to select terms to categorise longer documents". *International Journal of Computational Science and Engineering, ISSN 17427193, Volume 6, NO. 4, Pages 238-248*, 2001.

[41] S. Santander-Jimenez, M. Vega-Rodrguez. "Parallel Multiobjective Metaheuristics for Inferring Phylogenies on Multicore Clusters". *DOI 10.1109/T-PDS.2014.2325828, IEEE, Volume 26, NO. 6, Pages 1678-1692*, 2014.

[42] K.Bach, M.Lichman. "UCI machine learning repository". *April 30*, 2017.

[43] Monir Foqaha, Mohammed Awad. "Text Mining Using Radial Basis Function Neural Networks and Optimization Algorithms". *Master Thesis, Arab American University*, 2016.

[44] H. Zhao, G. Williams, J. Huang. "WSRF: An R Package for Classification with Scalable Weighted Subspace Random Forests". *Journal of Statistical Software, Volume 77, NO. 3, Pages 1-30*, 2017.

[45] C. Ling, X. Huang, H. Zhang. "AUC: a Better Measure than Accuracy in Comparing Learning Algorithms". *16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, June 1113, 2003, Proceedings.*

[46] Y. You, S. Song, F. Haohuan, A. Marquez, M. Dehnavi, K. Barker, K. Cameron, A. Randles, G.Yang. "MIC-SVM: Designing A Highly Efficient Support Vector Machine For Advanced Modern Multi-Core and Many-Core Architectures". *IEEE 28th International Parallel Distributed Processing Symposium, Pages 809-818*, 2014.

[47] W. Tian, T. Sevilla, W. Zuo. "A systematic evaluation of accelerating indoor airflow simulations using cross-platform parallel computing". *Journal of Building Performance Simulation, Volume 10, NO. 3, Pages 243-255*, 2017.

[48] A. Grandison, Y. Cavanagh, P. Lawrence, E. Galea. "Increasing the Simulation Performance of Large-Scale Evacuations Using Parallel Computing Techniques Based on Domain Decomposition". *Springer, Volume 53, Pages 1399-1438*, 2017.