Spring 5-18-2018

# Applications of Artificial Intelligence in Power Systems

Samin Rastgoufard
*srastgou*, srastgou@uno.edu

# Applications of Artificial Intelligence in Power Systems

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Engineering and Applied Science
Electrical Engineering

by

Samin Rastgoufard

B.S. Shiraz University, 2009
M.S. Shiraz University, 2012

May 2018

*To my lovely parents*
*and my sister*

# Acknowledgments

I would like to express my appreciation to my honorable advisor Dr. Dimitrios Charalampidis for all his guidance and support through this journey. His patience and encouragement were essential for the completion of this dissertation. I would like to express my gratitude to my dear uncle, Dr. Parviz Rastgoufard for his unconditional love and support and all the life lessons through these years. I would also like to thank Dr. Sumaiya Iqbal and Dr. MD. Tamjidul Hoque for their collaboration in some parts of this dissertation. I would like to thank my dear committee members Dr. Edit J. Kaminsky Bourgeois, Dr. Ittiphong Leevongwat, Dr. Emir Jose Macari, and Dr. Tumulesh Kumar S Solanky for their valuable comments and guidance.

My gratitude to my dad Mr. Kourosh Rastgoufard and my mom, Mrs. Farkhondeh Tajdaran, who are the loveliest and most supportive parents that anyone could wish. I like to thank my lovely sister Mrs. Sana Rastgoufard and my brother in law Dr. Ehsan Zahed for their unconditional love and emotional support.

Finally, I like to thank all of dear friends and family who helped me technically and emotionally to finish this dissertation.

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **ACOr** | Ant Colony Optimization for Continuous Domain |
| **AI** | Artificial Intelligence |
| **AMC** | AM-based Crossover |
| **ANN** | Artificial Neural Network |
| **ASPSO** | Analytical Selection Particle Swarm Optimization |
| **BGA** | Breeder Genetic Algorithm |
| **CART** | Classification and Regression Tree |
| **CCF** | Chromosome Correlation Factor |
| **CCR** | Correct Classification Rate |
| **DE** | Differential Evolution |
| **EC** | Evolutionary Computation |
| **ECOC** | Error-correcting Output Codes |
| **ED** | Economic Dispatch |
| **ERM** | Emperical Risk Minimization |
| **FNGA** | Fast Navigating Generic Algorithm |
| **FS** | Feature Selection |
| **GA** | Genetic Algorithm |
| **GS** | Grid Search |
| **HS** | Harmony Search |
| **ISDA** | Iterative Single Data Algorithm |
| **KDD** | Knowledge Discovery from Data |
| **KGA** | Kite Genetic Algorithm |
| **L1QP** | $L_1$ Soft-margin Quadratic Programming |
| **LOI** | Line Overload Index |
| **MFO** | Multi Fuel Option |
| **MLFNN** | Multilayer Feedforward Neural Network |
| **MLPN** | Multilayer Percepteron Network |
| **MOPSO** | Multi-Objective Particle Swarm Optimization |
| **MPSO** | Modified Particle Swarm Optimization |
| **MWV** | Max Wins Voting |
| **NR** | Newton Raphson |
| **NRLF** | Newton Raphson Load Flow |
| **OOB** | Out of Bag |
| **OVO** | One Versus One |
| **OVA** | One Versus All |
| **PAR** | Pitch Adjusting Rate |
| **PMU** | Phasor Measurement Units |
| **POZ** | Prohibited Operating Zones |
| **PSO** | Particle Swarm Optimization |

| | |
|---|---|
| **RBF** | Radial Basis Function |
| **RBFN** | Radial Basis Function Network |
| **RF** | Random Forest |
| **RMSE** | Root Mean Square Error |
| **SBS** | Sequential Backward Selection |
| **SCADA** | Supervisory Control and Data Acquisition |
| **SFS** | Sequential Forward Selection |
| **SMO** | Sequential Minimal Optimization |
| **SSA** | Social Spider Algorithm |
| **SSE** | Static Security Evaluation |
| **SSI** | Static Security Index |
| **STD** | Stand Deviation |
| **SVM** | Support Vector Machine |
| **SVR** | Support Vector Regression |
| **TB** | Tree Bagging |
| **TRGA** | Twin Removal Genetic Algorithm |
| **TS** | Tabu Search |
| **TSVR** | Tuned Support Vector Regression |
| **UC** | Unit Commitment |
| **UGA** | United Genetic Algorithm |
| **VDI** | Voltage Deviation Index |
| **VPE** | Valve Point Effect |

# Abstract

Artificial intelligence (AI) tools, which are fast, robust and adaptive can overcome the drawbacks of traditional solutions for several power systems problems. In this work, applications of AI techniques have been studied for solving two important problems in power systems.

The first problem is static security evaluation (SSE). The objective of SSE is to identify the contingencies in planning and operations of power systems. Numerical conventional solutions are time-consuming, computationally expensive, and are not suitable for online applications. SSE may be considered as a binary-classification, multi-classification or regression problem. In this work, multi-support vector machine is combined with several evolutionary computation algorithms, including particle swarm optimization (PSO), differential evolution, Ant colony optimization for the continuous domain, and harmony search techniques to solve the SSE. Moreover, support vector regression is combined with modified PSO with a proposed modification on the inertia weight in order to solve the SSE. Also, the correct accuracy of classification, the speed of training, and the final cost of using power equipment heavily depend on the selected input features. In this dissertation, multi-object PSO has been used to solve this problem. Furthermore, a multi-classifier voting scheme is proposed to get the final test output. The classifiers participating in the voting scheme include multi-SVM with different types of kernels and random forests with an adaptive number of trees. In short, the development and performance of different machine learning tools combined with evolutionary computation techniques have been studied to solve the online SSE. The performance of the proposed techniques is tested on several benchmark systems, namely the IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus power systems.

The second problem is the non-convex, nonlinear, and non-differentiable economic dispatch (ED) problem. The purpose of solving the ED is to improve the cost-effectiveness of power generation. To solve ED with multi-fuel options, prohibited operating zones, valve point effect, and transmission line losses, genetic algorithm (GA) variant-based methods, such as breeder GA, fast navigating GA, twin removal GA, kite GA, and United GA are used. The IEEE systems with 6-units, 10-units, and 15-units are used to study the efficiency of the algorithms.

**Keywords:** Aritifical intelligence (AI), static security evaluation (SSE), classification, regression, support vector machine/regression (SVM/R), random forest (RF), evolutionary computation (EC), modified particle swarm optimization (MPSO), multi-objective particle swarm optimization (MOPSO), feature selection (FS), economic dispatch (ED), genetic algorithm (GA)

# Chapter 1

# Introduction

## 1.1 The Necessity of this Research

Nowadays, recent developments and different new challenges in power systems dictate the need for several improvements in the power systems planning, operation, and control. The necessity of obtaining online solutions as fast as possible for different problems in power systems is becoming more apparent. Traditional methods are usually not able to solve power system problems in real time. In general, such methods are time-consuming and computationally expensive, and are not suitable for online monitoring and control. Artificial intelligence (AI) techniques, which include machine learning, data mining, and evolutionary (or meta-heuristic) computation methods, can successfully solve extremely challenging problems. Combining AI with traditional analytical techniques, such as statistical analysis, may also improve the overall performance of the techniques. The main advantages of AI tools are their speed, robustness, and relative insensitivity to noisy or missing data [1]. In this work, two important problems in power systems have been selected to be solved with different AI tools. The first is the static security evaluation (SSE) problem, and the second is economic dispatch (ED) problem. In the following sections, the details of each problem are discussed, and a brief review of existing works is presented.

## 1.2 Introduction to Applications of AI to Solve SSE

Nowadays, the growth of power systems in terms of both size and complexity indicates the importance of achieving a high degree of system security. At the same time, modern electric utilities are operating under stressed operating conditions and closer to their limits [2], [3] [5]. Therefore, it is clear that due to these higher demands a system may easily collapse under any moderate disturbance [6]. As a result, a fast, accurate online monitoring model is a necessity for preventing equipment damage, localized loss of power, loss of voltage, loss of frequency or, in severe cases, blackout [7], [8]. Security monitoring, contingency analysis, and security control are three major steps in security evaluation. In security monitoring, the operating conditions are reported to the operational engineer. In contingency analysis, the contingencies are screened and ranked in order of decreasing severity. In the security control step, control actions are implemented to return the insecure system to a secure state [9]. Security can be mainly classified as transient security and static security [10]. Transient security analysis studies the ability of the system to survive the transition to the steady state condition following a set of severe credible contingencies [11]. Monitoring the steady-state behavior of the system with regards to its ability to withstand credible contingencies is called SSE. SSE discovers any potential system overload or out-of-limit voltage. Contingencies include generating unit outages, transmission line outages, a sudden increase in demand, or loss of any equipment in the system. The contingencies with a high probability of occurrence are called credible contingencies. These are transmission equipment overloads and inadequate voltage levels at systems buses [12]. Contingency screening and ranking is a critical part of SSE requiring the solution of a large set of nonlinear algebraic equations for $N$ and $N-1$ system conditions [13]. Traditionally, SSE is solved offline due to significant computation time required. However, there is an increasing demand for more accurate and fast static assessment [14]. Supervisory control and data acquisition (SCADA) and phasor measurement units (PMU) have been used in power plant and transmission lines for a while. They are used to improve and control system reliability. However, even with a large amount of available data they cannot always take the suitable action to

prevent blackouts [8]. Performing online security analysis is hindered by the size and complexity of power systems. As mentioned above, SSE requires the solution of a large set of nonlinear algebraic equations [3]. It is then expected that in a real environment where the operating system conditions change regularly, it is crucial for SSE to be able to frequently and quickly assess the security level and determine appropriate preventive actions [2].

In recent years, the employment of AI tools to solve power system problems has been increasing. Combining machine learning, data mining, and evolutionary algorithms can be a powerful solution for real-time security assessment. These modern techniques have been shown to provide more robust results and, often, exhibit faster response times compared to traditional methods [2]. In some recent literature, different machine learning methods are used to solve SSE as a regression, binary classification, or multi-classification problem. Some of the most recently published papers are briefly mentioned below.

In [9], [13] and [15], different artificial neural networks (ANN) like radial basis function network (RBFN) and multilayer perceptron network (MLPN) have been used to solve SSE online as a binary classification problem or a regression problem. In [16], the SSE is solved as a binary classification problem by particle swarm optimization (PSO) and in [5] by PSO based $K$-means clustering. A technique combining a support vector machine (SVM) combined with different evolutionary algorithms for parameter tuning has been proposed in [2], [12], and [17] to solve multi-classification SSE online. In these works, the security regions are classified into four operating conditions. Different decision tree trainers have been compared with C-4.5 tree classifier in [18] to solve the binary SSE in electric power grid with the presence of PV power plants. Also, static security assessment and control of power systems using ANN techniques have been studied in [19], [20], [21], [22]. In [9] and [13], RBFN and multilayer feed-forward networks (MLFFN) were used for online static security assessment. In [23], ANNs were used for online contingency screening and ranking. However, most ANN-based techniques require a large number of neurons and hidden layers for achieving acceptable results. Training large ANNs can be a considerably time-consuming process, and may also result in model over-fitting.

In this work, several AI techniques have been used and proposed to solve and improve the performance of online SSE. These are explained briefly below.

First, multi-class support vector machine (SVM) with error correcting output codes (ECOC) (one-versus-one or OVO) is used to address the SSE classification problem. The performance of SVM heavily depends on its parameter selection. Therefore, several evolutionary (Meta-Heuristic) optimization techniques have been used to optimize SVM parameters. In particular, PSO, differential evolution (DE), Ant colony optimization for the continuous domain (ACOr) and harmony search (HS) are the four evolutionary computation (EC) techniques which were used to optimize the penalty parameter $C$ and radial basis function (RBF) kernel parameter ($\gamma$). For each method, the classification accuracy for each class and overall are presented. The techniques are also compared in terms of their training execution speed. For this part of the work, SSE is viewed as a 2-class, a 3-class or a 4-class classification problem, while commonly only 2 classes (secure or insecure) are considered. Moreover, additional optimization techniques are studied compared to previous works that used evolutionary optimization techniques to train SVMs. It will be demonstrated that all methods provide similar classification accuracy, while HS operates faster than other methods. An important conclusion is that the level of accuracy for each technique depends on the number of classes, namely, the number of security levels. The IEEE 39-bus has been used for implementing and validating the classifier performance. The proposed technique and its simulation results are presented in sections 5.1 and 6.2.1, respectively.

Second, support vector regression (SVR) is used to solve SSE as a regression problem. SVR is a powerful machine learning paradigm which attempts structural risk minimization rather than the empirical risk minimization adopted in ANN training. However, similarly to SVM, the performance of SVR heavily depends on its parameter selection. In [26] and [27], grid searching was used as an exhaustive method for parameter tuning, but due to the discretization of the search space, some information is lost. Traditional optimization methods may not perform satisfactorily in general, but evolutionary optimization methods such as genetic algorithm (GA), PSO and DE have been effectively used for tuning SVR

parameters [28]. In this work, an improved technique based on SVR and PSO, namely the tuned support vector regression by modified particle swarm optimization (TSVR-MPSO) is proposed for SSE. Recently, an approach was proposed for reliability prediction using SVR and PSO [24]. The main contribution in [24] is that different PSO particles are adapted using a different inertia weight based on an estimated global best. In this work, adaptation of the inertia weights is modified further, so that it is different for each one of the particle dimensions. In particular, in our proposed method, the PSO weight is updated for each particle based on the absolute distance between the global best and each particle's best position. The performance of TSVR-MPSO is compared with the method in [24] and with another weight-adapting PSO technique, MLFNN and RBFN methods. The IEEE 14-bus and 118-bus have been used for implementing and validating the regressor performance. The proposed technique and its simulation results are presented in sections 5.2 and 6.2.2 respectively.

Third, multi-objective PSO has been used for feature selection (FS) to improve the solution to the SSE problem. FS is an essential task for reducing the dimensionality and providing better performance of the classification algorithms [77]. Feature selection is a multi-objective problem with two conflicting objectives which are, first, maximizing the classification accuracy and, second, minimizing the number of features [77]. In previous works, feature selection for SSE has been solved without considering it to be a two-objective problem, and a predefined number of features has been set at the beginning. In this work, we are proposing the employment of multi-objective particle swarm optimization (MOPSO) for the purpose of selecting the best number of features and for improving the classification accuracy for SSE. The IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus systems have been used for the simulations. The proposed work is compared with the popular sequential feature selection (SFS) technique. The proposed method and the associated simulation results are presented in sections 5.3 and 6.2.3, respectively.

The fourth and final part of the dissertation work includes a voting scheme for solving the multi-classification SSE problem. Following FS by MOPSO, different machine

learning methods (9-classifier models) are trained. These 9 models are based on two well-known pattern classifiers. The first classifier is the SVM with different nonlinear kernels, such as the polynomial kernel and the RBF kernel having its parameters tuned by MPSO. The second classifier is a popular ensemble learning method, namely the random forest (RF), with an adaptive number of trees. The last stage of the technique involves a voting scheme which aims at improving the overall performance for online SSE. The IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus systems have been used for simulations. The proposed technique and the associated simulation results are presented in sections 5.4 and 6.2.4 respectively.

## 1.3 Introduction to Applications of EC to Solve ED

The massive consumption of fossil fuels has resulted in a dramatic reduction of these resources [4]. The power generation required for the operation of power systems is one of the leading causes of fossil fuel consumption. In order to reduce consumption by optimal usage of fuel resources, power should be generated at the lowest possible cost, while still meeting a known power demand and satisfying various constraints. This can be achieved by solving the economic dispatch (ED) problem which finds the best feasible power generation with minimum fuel cost while satisfying the generation constraints of the power units [29], [37]. Several classical and numerical methods, such as fast lambda iteration, Lagrange relaxation (LR), linear programming, and gradient methods have been traditionally used for solving the ED problem [38], [39], [40]. Often, these methods solve the ED problem by simplifying or ignoring some constraints such as the prohibited operating zones (POZ) of generators, the ramp rate limits [29], the valve-point effect (VPE) [41], and the multi-fuel options (MFO) [42]. Incorporating these real factors increases the complexity of the ED problem, which becomes a non-convex, non-continuous, and non-differentiable constrained optimization problem [29]. In general, traditional methods often fail to solve the non-simplified ED problem successfully.

Several evolutionary and metaheuristic algorithms have been used in the literature to solve the ED problem and to overcome the difficulties of conventional optimization

methods. Some of the advanced evolutionary algorithms used include GA [41], [43], tabu search (TS) [44], PSO [29], differential evolution (DE) [45], ant colony optimization (ACO) [46], harmony search [47], artificial bee colony (ABC) [48], and the social spider algorithm (SSA) [49], [50]. Furthermore, hybrid algorithms have also been employed to solve the ED problem [51, 52, 53].

In this work, advanced variants of GAs [54] are used to solve the ED problem. The idea of GA was inspired by Darwin's theory of evolution and was first invented by John Holland [55]. GA, in its implementation, starts with a set of individuals (initial population of candidate solutions) that are evolved over consecutive generations (epochs) through selection and variation to solve an optimization problem. In GA, individual problem-solutions, to which the values of the solution variables are encoded, are referred to as chromosomes. GA evolves through the natural adaptation process in which the fitter chromosomes tend to survive, breed, and propagate their genetic information to the future generations. GA variants, with integrated advanced and innovative strategies, help produce competitive solutions. Though the GA variants were separately shown, a solution-suite can be easily formulated to have the combined benefits of the GA variants. The GA variants which are used in this work are breeder GA (BGA), fast navigating GA (FNGA), twin removal GA (TRGA), kite GA (KGA), and united GA (UGA). Three IEEE benchmark test systems with 6-unit, 15-unit, and 10-unit are used to study the efficacy of the proposed algorithms. The 6-unit and 15-unit have POZ, ramp rate limits and transmission line losses as their constraints, while the 10-unit system is the multi-fuel system with valve-point effect. In section 6.2.5, the results obtained by different advanced GA variants are compared for solving the ED problem. It is also shown that for some case studies the kite GA (KGA) and twin removal GA (TRGA) outperform the results obtained by some recently proposed evolutionary and metaheuristic techniques, such as the modified social spider algorithm (MSSA) [50] and the backtracking search algorithm BSA [42].

## 1.4 Overview of the Dissertation Chapters

In this section, the organization of the following chapters is presented. In chapter 2, two power system problems (namely SSE and ED) are described. In chapter 3, an overview of different machine learning techniques and data mining tools which have been used in this dissertation are presented. These include SVM, SVR, RF, RBFN, MLFNN, and FS. In chapter 4, several EC methods are explained shortly. These methods are PSO, the variants of GA (BGA, FNGA, TRGA, KGA, and UGA), DE, ACOr, and HS. Proposed methods and implementation steps are described in section 5. The proposed techniques are parameter selection of multi-class SVM with EC methods for online SSE, tuned support vector regression by modified PSO for online SSE, feature selection by multi-objective PSO for SSE, multi-classifier voter model for SSE, and genetic algorithm variant-based effective solutions for ED. In chapter 6, case study and simulation results of each proposed method are presented. Finally, chapter 7 closes with some conclusion and suggestions for future work.

# Chapter 2

# Power Systems Problems

In the following sections, an overview of two power system problems is presented. The first problem is static security evaluation (SSE) and the second problem is economic dispatch (ED).

## 2.1 Static Security Evaluation

The security of power systems is an important issue in online power system networks. In order to prohibit a blackout, all power system equipment should work within their appointed limits [12]. The main goal of security assessment is to evaluate the robustness of the system or the security level of the system during and after an contingency incident. The security of the system should be analyzed online due to the influence of different contingencies and the time-to-time variation of system operating conditions. Traditionally, SSE is solved using the algebraic load flow equations for any outage, one at a time, which requires a huge number of computations. This is why conventional methods cannot assess the security level of the power systems in real-time. In this work, the traditional Newton Raphson (NR) load flow has been only used off-line in order to generate the data required for the purpose of training a classification or regression system. Once the system is trained, the SSE can be performed in real-time.

In power systems, examples of contingencies include an outage of a line or generator, a sudden increase in load, or a three-phase fault in the system. The set of contingencies

which are considered in this work were one-line outages, namely $N-1$ contingencies, and random changes in the loads from 80% to 120% of their base values. In literature, the system is considered static secure if it remains in steady state after a contingency occurrence or, in other words, if the MVA flow of the branches and the amplitude of bus voltages stay within their specified limits. Eq. 2.1 and Eq. 2.2 show the voltage magnitude limit and the maximum MVA flow respectively.

$$|V_k^{min}| \leq |V_k| \leq |V_k^{max}| \quad k = 1, 2, ..., N_{bus} \tag{2.1}$$

$$S_{km} \leq S_{km}^{max} \tag{2.2}$$

where $|V_k^{min}|$, $|V_k^{max}|$ and $|V_k|$ are the minimum voltage limit, maximum voltage limit, and the bus magnitude of bus $k$. Moreover, $S_{km}$ and $S_{km}^{max}$ are the MVA flow and the MVA limit of the branch from bus $k$ to bus $m$.

In order to obtain the MVA flow and bus voltage magnitude, the nonlinear load flow equation should be solved. Different static security index (SSI) types have been presented in the literature to determine the level of system security. Some SSIs have been proposed to indicate overloaded lines or bus voltages that deviate from the normal operation limits [2], [12]. The SSI can be calculated by the line overload index (LOI) and voltage deviation index (VDI) which are presented in Eq. 2.3 and Eq. 2.4. The SSI index is defined in Eq. 2.5:

$$LOI_{km} = \begin{cases} \frac{S_{km} - S_{km}^{max}}{S_{km}^{max}} \times 100 & S_{km} > S_{km}^{max} \\ 0 & S_{km} \leq S_{km}^{max} \end{cases} \tag{2.3}$$

$$VDI_k = \begin{cases} \frac{|V_k^{min}| - |V_k|}{|V_k^{min}|} \times 100 & |V_k| < |V_k^{min}| \\ \frac{|V_k| - |V_k^{max}|}{|V_k^{max}|} \times 100 & |V_k| > |V_k^{max}| \\ 0 & |V_k^{min}| < |V_k| < |V_k^{max}| \end{cases} \tag{2.4}$$

$$SSI = \frac{w_1 \sum_{i=1}^{N_{Line}} LOI_i + w_2 \sum_{i=1}^{N_{Bus}} VDI_i}{N_{Line} + N_{Bus}} \tag{2.5}$$

where $w_1$ and $w_2$ are the constant weighting factors, which can be changed according to one's preference. Moreover, $N_{Line}$ and $N_{Bus}$ are the total number of lines and buses of the system, respectively.

The SSE can be solved as a classification or regression problem. The SSI index will be used as the output of the training model, if SSE is considered as a regression problem. To solve SSE as a classification problem, the 2-class, 3-class, and 4-class labeling are presented in Table 2.1, Table 2.2, and Table 2.3, respectively, for classifying the security level of the system.

**Table 2.1: SSI labeling for two classes**

| Static Security Index (SSI) | Class Label |
|---|---|
| $SSI \leq 5$ | Class 1: Secure |
| $SSI > 5$ | Class 2: Insecure |

**Table 2.2: SSI labeling for three classes**

| Static Security Index (SSI) | Class Label |
|---|---|
| $SSI \leq 1$ | Class 1: Safe |
| $SSI > 1$ & $SSI \leq 15$ | Class 2: Alarm |
| $SSI > 15$ | Class 3: Emergency |

**Table 2.3: SSI labeling for four classes**

| Static Security Index (SSI) | Class Label |
|---|---|
| $SSI \leq 1$ | Class 1: Safe |
| $SSI > 1$ & $SSI \leq 5$ | Class 2: Alarm |
| $SSI > 5$ & $SSI \leq 15$ | Class 3: Insecure |
| $SSI > 15$ | Class 4: Emergency |

## 2.2 Economic Dispatch

Economic dispatch (ED) is one of the important problems in power system control and operation. ED is a sub-problem of unit commitment (UC) as well as a nonlinear optimiza-

tion problem with various constraints. The objective of solving the ED problem is to find the optimum power generated by all generators in a power system in order to minimize the total fuel cost of the system. At the same time, a number of constraints such as load demand, spinning reverse capacity, ramp rate limits, and generator prohibited operating zones need to be satisfied. The ED objective function which needs to be minimized can be expressed as follows:

$$\sum_{j=1}^{n} F_j(P_j) = \sum_{j=1}^{n} a_j + b_j P_j + c_j P_j^2 \tag{2.6}$$

where, $F_j(P_j)$ is the fuel cost function of the $j^{th}$ power unit, $P_j$, and $n$ is the total number of power units. The parameters $a$, $b$, $c$ represent constant coefficients associated with the fuel cost. In practice, the valve-point effect (VPE) of steam-power plants exhibits ripples which can be modeled as a recurring rectified sinusoid. Therefore, the VPE effect can be incorporated in the ED objective function by modifying $F_j(P_j)$ as follows:

$$F_j(P_j) = a_j + b_j P_j + c_j P_j^2 + |e_j sin(f_j(P_j^{min}) - P_j| \tag{2.7}$$

where $P_j^{min}$ is the minimum power generated by the $j^{th}$ power unit. Moreover, $e_j$ and $f_j$ are constant coefficients describing the VPE [41]. Furthermore, some power units can operate using multiple fuels with different associated costs [49]. Based on the power generation requirements, the fuel with minimum cost should be selected for each unit. Consequently, the fuel cost function of the $j^{th}$ power unit can be modified to incorporate the multi-fuel option as follows:

$$\begin{aligned} F_j(P_j) = \min_{k=1,\ldots,m} (a_{j,k} + b_{j,k} P_j + c_{j,k} P_j^2 \\ + |e_{j,k} sin(f_{j,k}(P_j^{min}) - P_j|) \end{aligned} \tag{2.8}$$

where, $m$ is the total number of fuel options while $a_{j,k}, b_{j,k}, c_{j,k}, e_{j,k}$ and $f_{j,k}$ are the fuel cost coefficients of the $j^{th}$ power unit's $k^{th}$ fuel option [42]. In addition to defining the ED objective function, certain constraints should be considered. First, the total generated

power has to be equal to the power demand plus any power losses. The requirement to maintain the active power balance of the system can be expressed as follows:

$$\sum_{j=1}^{n} P_j = P_D + P_L \tag{2.9}$$

where $P_D$ is the load demand and $P_L$ is the line loss. In particular, $P_L$ can be calculated by:

$$P_L = \sum_{i=1}^{n}\sum_{j=1}^{n} P_j B_{ij} P_j + \sum_{i=1}^{n} B_{i0} P_i + B_{00} \tag{2.10}$$

where $B_{ij}$ are the line loss coefficients [29]. Each power unit may be restricted by additional constraints depending on the problem at hand. In particular, the power generated by each power unit may not drop below a minimum value, $P_j^{min}$, or exceed a maximum value, $P_j^{max}$. Moreover, the ramp rate limits may restrict the power generated by the $j^{th}$ unit as follows:

$$max(P_j^{min}, P_{j0} - LR_j) \leq P_j \leq min(P_j^{max}, P_{j0} + UR_j) \tag{2.11}$$

where $P_{j0}$ is the previous interval output power, while $LR_j$ and $UR_j$ are the lower and upper ramp rate limits of the $j^{th}$ unit, respectively. Another constraint which is considered in different works is the generator prohibited operating zone (POZ). The POZ represents an infeasible or forbidden operation range of a power unit. Eq. 2.12 introduces this additional constraint considering $z_j$ POZs for the $j^{th}$ power unit.

$$P_j \in [P_j^{min}, P_j^{l,1}] \bigcup [P_j^{u,1}, P_j^{l,2}] \bigcup ... \bigcup [P_j^{u,z_j}, P_j^{max}] \tag{2.12}$$

where $P_j^{l,k}$ and $P_j^{u,k}$ are, respectively, the lower and upper bounds of the $j^{th}$ power unit's $k^{th}$ POZ and $z$ is the total number of POZs [29, 49].

# Chapter 3

# Machine Learning and Data Mining

## 3.1 Introduction

It is well-known that during the last few decades vast amounts of information are continuously generated around the globe. Therefore, it does not come as a surprise that large amounts of data are available in all engineering and science fields, especially considering the nature of these fields. An important objective of Engineering and science is to observe and analyze various natural phenomena, as well as man-made systems. For this purpose, given the huge amount of data available, it is important to be able to identify and extract only this information which is useful for a particular application. Data mining is a subfield of knowledge discovery from data (KDD), which aims at processing the data in order to identify patterns and to present best relationships among data. Data mining includes data preprocessing, feature selection (FS), classification, clustering, regression, data warehousing, frequent pattern mining, and outlier analysis.

Machine learning is a sub-field of AI. It mainly concentrates on proposing algorithms that can learn from data, and that can draw conclusions about the data. It is then clear that machine learning algorithms can be used in data mining for the purpose of training models, so that necessary information can be extracted from the data. Artificial neural networks (ANN), such as the radial basis function network (RBFN) [30] and multilayer feedforward neural network (MLFNN) [31], as well as fuzzy logic [32], support vector machines (SVM) [33], decision trees, random forest (RF) [34], $k$-means [35], and

reinforcement learning [36] are some of the available machine learning tools.

In this chapter, a brief description of some machine learning and data mining methods, especially the ones which were used to solve the SSE problem, is presented. In particular, SVM and SVR are discussed in section 3.2, RF is described shortly in section 3.3, and the RBFN and MLFNN techniques are presented in section 3.5 and 3.4. Finally, FS is outlined in section 3.6.

## 3.2 Support Vector Machine

The SVM was first introduced by Cortes and Vapnik in 1995 for supervised binary classification [33]. However, it was later extended to solve multi-class and regression problems as well. In general, SVM possesses two special properties. First, it is capable of maximizing the margin of separation between different classes. Second, it supports nonlinear functions by utilizing different kernels [58]. The SVM constructs an optimal hyperplane classifier that classifies the data without error by maximizing the margin of separation, which in turn minimizes the empirical risk (the average loss of an estimator for a finite set of data) and expected risks (hypothesis value of loss function). Empirical risk minimization (ERM) minimizes the error on the training data and is used to give theoretical bounds on SVM performance. The idea of risk minimization is not only measure the performance of an estimator by its risk, but to actually search for the estimator that minimizes risk over distribution [59].

A constrained quadratic programming problem has to be solved to obtain the optimal hyperplane. This may be a linear or nonlinear combination of support vectors (a subset of training data) [58]. In order to construct the optimal hyperplane, the SVM classifier solves the following optimization problem if the two classes are linearly separable:

$$\min_{w^{ij},b^{ij},\zeta_t^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \zeta_t^{ij} \tag{3.1}$$

$$(w^{ij})^T x_t + b^{ij} \geq 1 - \zeta_t^{ij} \quad if \quad y_t = i \tag{3.2}$$

$$(w^{ij})^T x_t + b^{ij} \geq -1 + \zeta_t^{ij} \quad if \quad y_t = j \tag{3.3}$$

$$\zeta_t^{ij} \geq 0 \tag{3.4}$$

where $w^{ij}$ is the weight vector for the hyperplane, $C$ is the penalty parameter, $\zeta_t^{ij}$ is a slack variable. For better illustration, Fig. 3.1 shows binary SVM linear classification. This Fig has been generated in Matlab.

When two classes are not linearly separable, following optimization problem should solve, as:

$$\min_{w^{ij},b^{ij},\zeta_t^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \zeta_t^{ij} \tag{3.5}$$

$$(w^{ij})^T\phi(x_t) + b^{ij} \geq 1 - \zeta_t^{ij} \quad if \quad y_t = i \tag{3.6}$$

$$(w^{ij})^T\phi(x_t) + b^{ij} \geq -1 + \zeta_t^{ij} \quad if \quad y_t = j \tag{3.7}$$

$$\zeta_t^{ij} \geq 0 \tag{3.8}$$

where $\phi(x_t)$ is a nonlinear kernel function. Mainly, two types of kernel functions are used for nonlinear classification in SVM. The Gaussian or RBF is the first and, possibly, best choice owing to its capability of successfully handling nonlinear relations and to its accuracy. The other well-known and easy to use kernel is the polynomial which can be of different order, $q$. Fig. 3.2 which has been created in Matlab illustrates an example of nonlinear (RBF) binary SVM classification.

Different types of solvers can be used to solve the SVM optimization problem. $L_1$ soft-margin quadratic programming (L1QP) [61] minimization is one of the mostly used optimization solvers. Other than L1QP, iterative single data algorithm (ISDA) [62], and sequential minimal optimization (SMO) [63] are other solvers who have been implemented and compared in this work. In this work, we used both RBF and polynomial kernel for multi-classification. However, the performance of SVM with RBF kernel heavily depends on its parameters tuning. Therefore, different EC algorithms are used to tune the penalty parameter, $C$, and the RBF kernel parameter, $\gamma$.

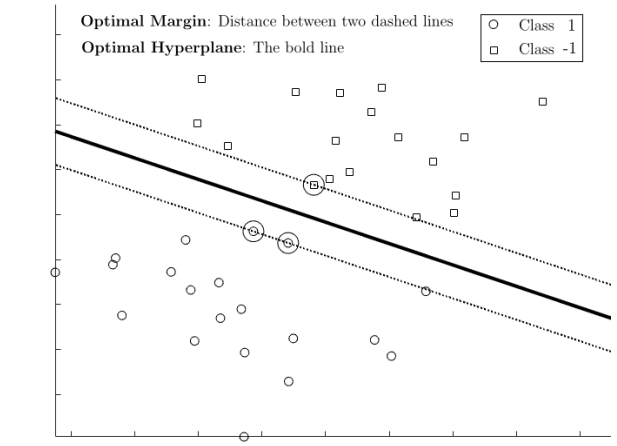


**Figure 3.1: Linear binary SVM classifier**

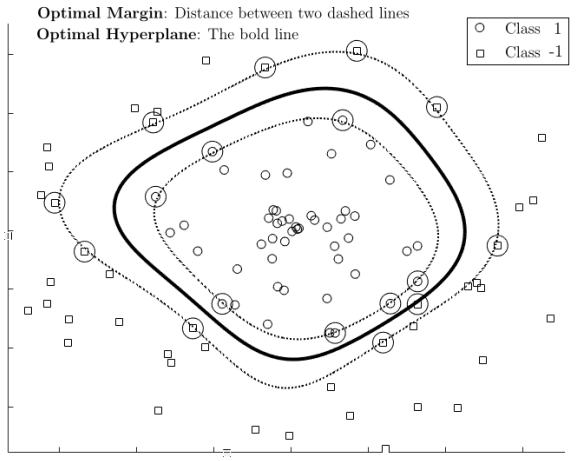


**Figure 3.2: Nonlinear binary SVM classifier**

### 3.2.1 Multi-Class Support Vector Machine

Binary SVM classifiers can be extended to multi-class classifiers. For solving the multi-class problem, a combination of several binary SVM classifiers should be trained. One-versus-all (OVA), one-versus-one (OVO), dense random, and sparse random are some of the multi-class SVM coding strategies [2]. In OVA coding, SVM constructs $k$ different distinct classes, considering two classes, namely class $i$ and all other classes. The OVA technique is simple, but it is computationally expensive. Based on the OVO technique (which is also called pairwise SVM), a total of $k(k-1)/2$ binary classifiers are constructed. They are trained by data belonging to the corresponding two classes only [12]. Classification based on the OVO method uses the max-wins voting (MWV) strategy to pick the class with the largest number of votes. In dense random, each binary learner is assigned

**Table 3.1: OVO coding scheme**

| | Learner 1 | Learner 2 | Learner 3 |
|---|---|---|---|
| Class 1 | 1 | 1 | 0 |
| Class 2 | -1 | 0 | 1 |
| Class 3 | 0 | -1 | -1 |

randomly to a "positive class" or a "negative class". In sparse random, each binary learner is again assigned randomly to a positive or negative class, but with probability 0.25 for each class, and by ignoring both classes with probability 0.5 [64]. In this work, the OVO method has been used to extend the binary SVM to the 3-class and 4-class classification problems.

In [64], error-correcting output codes (ECOC) with coding and decoding steps are proposed for solving multi-class classification. This approach improves classification accuracy compared to other multi-class models. The ECOC model reduces the classification problem with 3 or more classes to a set of 2-class classification problem. The coding design of ECOC determines how the 2-class learners are trained. The decoding design of ECOC decides how to compress the predictions of the 2-class classifiers. For better illustration, suppose that a 3-class problem using the OVO coding strategy and loss $g$ decoding scheme wants to be solved with SVM learner. An OVO coding design is presented in Table 3.1. Learner 1 trains class 1 and class 2 as class positive and class negative respectively. Rows containing a 0 infer to the 2-class learner to ignore all observations in the corresponding classes. Other learners are trained similarly. Assume that the coding design is matrix $M$ with elements $m_{kl}$. A new observation is assigned to the class $\hat{k}$ that minimizes the aggregation of the losses for the $L$ binary learners. Eq. 3.9 shows the observation for class $\hat{k}$ which minimizes the aggregation [64].

$$\hat{k} = \underset{k}{argmin} \frac{\sum_{l=1}^{N} |m_{kl}| g(m_{kl}, s_l)}{\sum_{l=1}^{N} |m_{kl}|} \tag{3.9}$$

where $L$ is the total number of learners and $s_l$ is the predicted classification score of the class positive of learner $l$.

### 3.2.2 Support Vector Regression

Support vector regression (SVR) is an extension of SVM to accommodate regression problems. The objective of SVR is to estimate a function which maps $l$-dimensional input vectors, $x_i \in R^l$, to real-valued outputs, $y_i$, $i = 1, ..., N$, where $N$ is the number of data patterns. For the linear case, SVR can be described as follows [24, 65, 67]:

$$y_i = f(x_i) = w^T x_i + b \tag{3.10}$$

where $w$ and $b$ are the weight vector and the intercept of the regression model, respectively. The values of $w$ and $b$ should be obtained based on the available dataset to determine the optimal linear function. This linear case can be extended to nonlinear mappings as $\Phi : R^l \to S$, where $S$ is the feature space of $\Phi$. The nonlinear case can be formulated as:

$$y_i = f(x_i) = w^T \Phi(x_i) + b \tag{3.11}$$

In $\varepsilon - SV$ regression, the goal is to find a function $f(x_i)$ which does not deviate more than $\varepsilon$ from the targets, $y_i$, while concurrently remain as flat as possible. The latter implies minimizing the Euclidean norm of the linear weight, namely $\|w\|^2$. For the best regression function, the soft margin $\varepsilon$-insensitive loss function is defined as:

$$|\zeta|_\varepsilon = \begin{cases} 0, & \text{if } |\zeta| < \varepsilon \\ |\zeta| - \varepsilon, & \text{otherwise.} \end{cases} \tag{3.12}$$

The deviation of the training data outside the $\varepsilon$-insensitive zone can be specified by slack variables $\zeta, \zeta^*$ which are used for minimizing the empirical risk. Figure 3.3 from [67] shows the soft margin loss setting for a linear SVR.

The formulation stated by Vapnik [61] leads to solving the following quadratic optimization problem:

$$\min J(w, \zeta, \zeta^*) = \frac{1}{2}\|w\|^2 + C \sum_{i=0}^{n} (\zeta + \zeta^*) \tag{3.13}$$

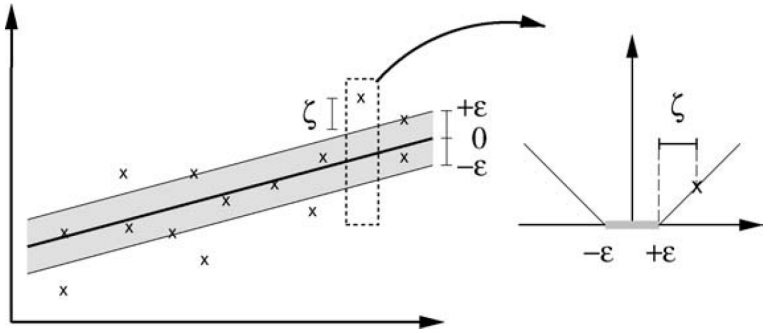


**Figure 3.3: The soft margin loss setting for a linear SVR [67]**

$$s.t. \begin{cases} y_i - w^T\Phi(x_i) - b \leq \varepsilon + \zeta_i \\ w^T\Phi(x_i) - b - y_i \leq \varepsilon + \zeta_i^* \quad i = 1, 2, ..., n \\ \zeta_i, \zeta_i^* \geq 0 \end{cases} \tag{3.14}$$

where $C > 0$ is a penalty coefficient which determines the trade-off between empirical and generalization errors [66]. The optimization problem in Eq. 3.13 can be solved in the standard dual method utilizing Lagrange multipliers. The solution of this quadratic optimization problem is:

$$f(x_i) = \sum_{j=0}^{n} (\alpha_i - \alpha_i^*) K(x_i, x_j) + b \tag{3.15}$$

where $\alpha_i, \alpha_i^*$ are Lagrange multipliers, and $K(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$ is the kernel function. In this paper, the radial basis function is used as the kernel function:

$$K(x_i, x_j) = exp(\frac{-\gamma^2}{2}|x_i - x_j|^2), \gamma \in R \tag{3.16}$$

For getting best regression results, the $\gamma$ (width of RBF), $\varepsilon$, and $C$ parameters have to be properly tuned. Figure 3.4 which is generated in Matlab, shows an example which presents the optimal margin and optimal hyperplane of nonlinear SVR.

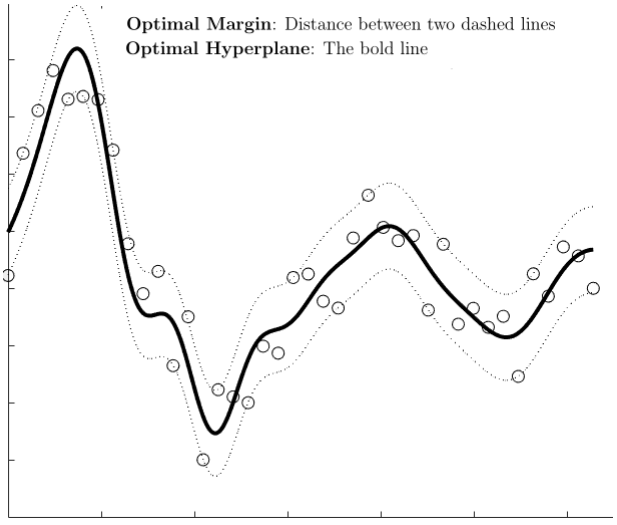


**Figure 3.4: Nonlinear SVR**

## 3.3 Random Forest

RF is an ensemble learning method used for classification and regression. An ensemble method is formed when a set of weak learners come together to form a strong learner. RF works by building a group of decision trees [34]. In the case of classification, the RF output is obtained as the most popular response considering all trees. In the case of regression, the output is the mean of prediction associated with the individual trees [68]. In RF, except for tree bagging [69], the processes of finding the root node and splitting the feature nodes are random. The RF is easy to use, and is robust to noise. Moreover, the RF mitigates the overfitting issue of decision trees during training, thus improving the generalization performance [70]. In [71], a collection of decision trees with controlled variance is used to construct the RF. Each tree depends on the values of a random vector which is sampled independently and distributed alike in the forest. In [71], uncorrelated trees using a classification and regression tree (CART) are combined with randomized node optimization and bagging technique in order to build the forest. Out of bag (OOB) error has been used as an estimate of the generalization error.

In RF a random independent vector $\phi_i$ is generated for the $i^{th}$ tree with the same distribution of past random vectors. A tree is using the training set and $\phi_i$ to grow. In short, RF is a classifier which consists of several tree classifiers, $C(x, \phi_i),\ i = 1, ..., n$, where $\phi_i$ is an independently distributed random vector. Each tree builds a unit vote for

the most popular class for input $x$ [71].

The general technique of bootstrap aggregating or tree bagging (TB) is being used to train the RF. In this algorithm, for a number of trees, $N$, random cases with the replacement of the training set is selected. At each node, $m$ predictor variables are selected randomly, and the one with the best split is applied for the binary split on the node. This process should be repeated for each node. By taking the majority vote among the outputs of all the individual trees, the prediction can be checked for the test data. The optimal number of trees is found by the OOB error, which is the mean prediction error on each training sample.

## 3.4 Multi-Layer Feedforward Neural Nework

The feedforward neural network (FNN) is the first and simplest type of ANN. There is no cycle or loop in this ANN, and the information moves in only one direction, namely forward, from the input nodes, through the hidden nodes, and finally to the output nodes [31]. In MLFNN, each hidden layer can be considered as a single output perceptron network, and the output layer is essentially a soft thresholded linear or nonlinear combination of the hidden layers. Any kind of input-output mapping can be modeled by MLFNN with enough neurons in the hidden layers. The sigmoidal activation function may be used in hidden layers as:

$$Y(x) = \frac{1}{1 + e^{-x}} \tag{3.17}$$

Each layer is attached to the previous layer with some weights [9]. Backpropagation algorithms (BPA) are used for training MLFNN using parameters such as the momentum factor $\alpha$, and the learning rate $\eta$. If $\eta$ is set to be small, the learning rate is also small, but if it set to a large value, training may become unstable. A momentum factor $\alpha$ can be added to increase the value of the rate $\eta$ without making the process unstable. The weights associated with the connections between the hidden layer and the output layer

are updated as follows:

$$w_b(j,k,t+1) = w_b(j,k,t) + \eta\delta_k(t)Y_b(j) + \alpha(w_b(j,k,t) - w_b(j,k,t-1)) \tag{3.18}$$

where $j$ varies from 1 to the total number of hidden layers, $N_h$, and $k$, varies from 1 to the total number of neurons in the output layer, $N_k$. Moreover, $Y_b(j)$ is the output from the hidden layers. The weights associated with the connections between the hidden layer and the input layer are updated as follows:

$$w_b(j,k,t+1) = w_b(j,k,t) + \eta\delta_j(t)Y_a(i) + \alpha(w_a(i,j,t) - w_a(i,j,t-1)) \tag{3.19}$$

where $i$ varies from 1 to the number of inputs to the network, $N_i$, $Y_a(i)$ is the output of the first layer, and $\delta_j(t)$ is the error corresponding to the $j^{th}$ output after the $t^{th}$ iteration.

The errors $\delta_k(t)$ and $\delta_j(t)$ are related as follows:

$$\delta_j(t) = \sum_{k=1}^{K} \delta_k(t) w_b(j,k,t) \tag{3.20}$$

The measn square error (MSE) for the training patterns after the $t^{th}$ iteration is given by:

$$MSE(t) = (\frac{1}{N_p}) \sum_{p=1}^{N_p} (X_{1p} - X_{2p}(t))^2 \tag{3.21}$$

where $N_p$ is number of patterns in the training set. Training stops when the maximum number of iterations is reached, or when an acceptable MSE values is obtained. The MLFNN model from [9] is shown in Fig. 3.5.

## 3.5 Radial Basis Function Network

The RBFN has been used in nonlinear function approximation, time series prediction, and classification. Commonly, RBFN has three layers: an input layer, a hidden layer with a non-linear RBF activation function, and a linear output layer. An input vector, $X$, is used as input to all RBFs, each with different parameters and the output of the network,

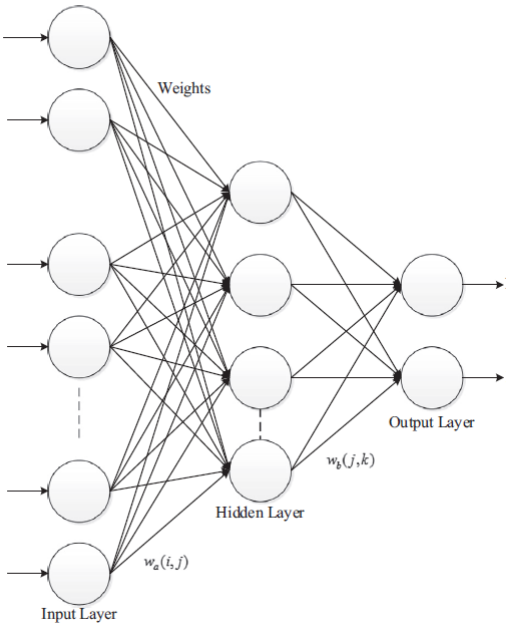


**Figure 3.5: Structure of MLFNN [9]**

$Y$, is a linear combination of the outputs from RBFs in the hidden layer [30, 74, 75]. The output of the network is:

$$Y(X) = \sum_{i=1}^{N} w_j.exp(\frac{-||X - C_i||^2}{\gamma}) \tag{3.22}$$

$N$, is the number of neurons in the hidden layer, $C_i$ is the center vector for neuron $i$, $w_j$ is the weight of neuron $i$ in the linear output neuron, and $\gamma$ is the RBF kernel parameter. The weights can be derived in a manner that the fit between output and the input is optimized. The structure of the RBF from [76]is presented in Figure 3.6. In the basic form of the RBFN, all inputs are connected to each hidden neuron. RBFNs are universal approximators, in the sense that given a sufficient number of hidden neurons they can approximate any continuous function. Selecting an appropriate number of neurons is important because a small number of neurons will result in low function classification accuracy. On the other hand, a large number of neurons may cause overfitting of the input data, which may deteriorate the global generalization performance.

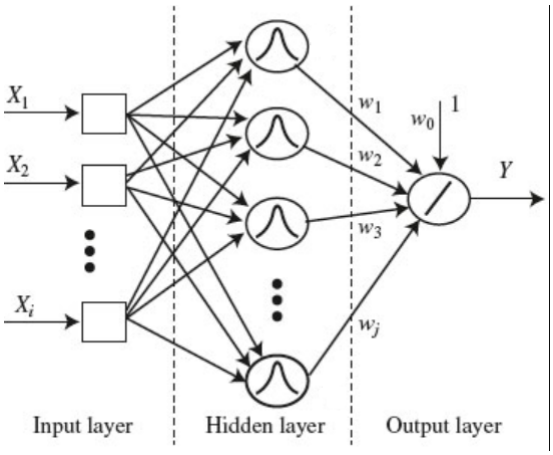


**Figure 3.6: Structure of RBFN [76]**

## 3.6 Feature Selection

Data often consists of a large number of features, many of which may be redundant. For instance, features may be highly correlated with other. Feature selection (FS) is the process of selecting only a subset of necessary features from the original data. A Large number of features may reduce the performance of data mining and machine learning [77], while FS improves the speed and generalization performance of classifiers [79].

FS algorithms belong in the category of dimensionality reduction methods. However, as opposed to other dimensionality reduction methods, FS selects a subset of the existing features without applying any transformation to the data. In other words, FS attempts to find a feature subset, $Y_m$, from a feature set, $X = x_1, x_2, \ldots, x_n$, where $m < n$, so that an objective function is minimized (or maximized). For a dataset with $n$ features, the total number of possible subsets is $2^n$. Apparently, the problem can become significantly complex for large $n$ [78].

FS algorithms mainly consist of filter or wrapper approaches. Filter approaches evaluate subsets of some information such as interclass distance, statistical dependence, or information-theoretic measures. Filter methods exhibit a fast execution time and generality. However, they force to select an optional number of features to be selected. On the other hand, wrapper approaches as opposed to filter methods, use a learning algorithm, such as a classifier or a regression learner, and the performance of the learner is the objective used for selecting the features. Wrapper techniques are more accurate comparing

to filter methods. However, their execution is slow, and they lack generality.

It should be mentioned that dimensionality reduction algorithms which combine features corresponding to different dimensions are not appropriate for SSE. For SSE, the data is obtained through phase measurement units (PMUs) installed at substations and power plants. The goal is to install a small number of PMUs to only acquire these features required for classification. This is possible when FS is used, since PMUs only have to be placed at $m$ appropriate locations, namely the ones associated with the selected feature subset. On the other hand, methods which use feature combinations require that all $n$ original features are available in order to produce the smaller number of combined features. This implies that PMUs would have to be installed at all $n$ locations.

Most works presented in the SSE literature use only the classifier's performance has been considered as the objective of FS. Such an FS example is sequential forward selection (SFS). Yet, a second objective should be minimizing the number of features. The FS method proposed in this work is a two-objective problem which attempts to minimize both the classification error rate and the number of features, which are often two conflicting objectives.

In this work, sequential forward selection (SFS) and multi-objective particle swarm optimization (MOPSO) (*see* section 4.2.1) have been used to select appropriate features for the SSE problem. SFS is explained briefly in section 3.7. The proposed technique which uses MOPSO for FS is discussed in section 5.3.

## 3.7 Sequential Forward Selection

SFS is a bottom-up search procedure. It first initializes an empty feature subset. Then, the selected features are gradually added to the subset, from the original feature set, based on a fitness function which aims at minimizing the MSE. In each iteration, the new feature to be included in the subset is selected among the remaining available features of the feature set. The objective is that the updated subset should produce a smaller classification error with respect to the subset which would be formed by the addition of any other feature. SFS performs best when the optimal subset has a small number of

features. The advantages of SFS are its simplicity and speed, and this the reason why many applications have been proposed based on SFS [73]. The main disadvantage of SFS is that it is unable to remove features that do not positively contribute to the MSE after the addition of other features.

# Chapter 4

# Evolutionary Computation

Evolutionary computation (EC) is a subfield of artificial intelligence (AI) which has mainly adopted Darwinian principles and population-based optimization processes inspired by biological evolution. The EC methods can solve optimization problems for a wide range of applications. The trial-and-error and randomness-based techniques employed by EC techniques facilitate the avoidance of local minima [1]. In EC, a random initial set of solutions (i.e., the first generation) is first created. Subsequently, new generations are iteratively updated by removing the less *fit* solutions (selection), by combining solutions (recombination), and by adding small random changes to members of the population (mutation). The population evolves while improving the fitness of its members. In other words, EC techniques are mainly based on the recombination between population members and the mutation of individual members to obtain the necessary diversity, and on selection, which increases the quality of members in future generations [80].

Swarm intelligence (SI), a subfield of EC, has been used in several applications. SI mimics the behavior of bird flocking or swarms of insects, such as ants, which are interacting locally with each other and with their environment. In general, there is no centralized control method or intelligence in a particular member, but the interactions between the members lead to the sharing of intelligence within the population [81]. Thus, the swarm exhibits a strong collaboration, such as communication, exchanging, and flowing of information between its members, which is a key to success of the swarm intelligence (Fig. 4.1).

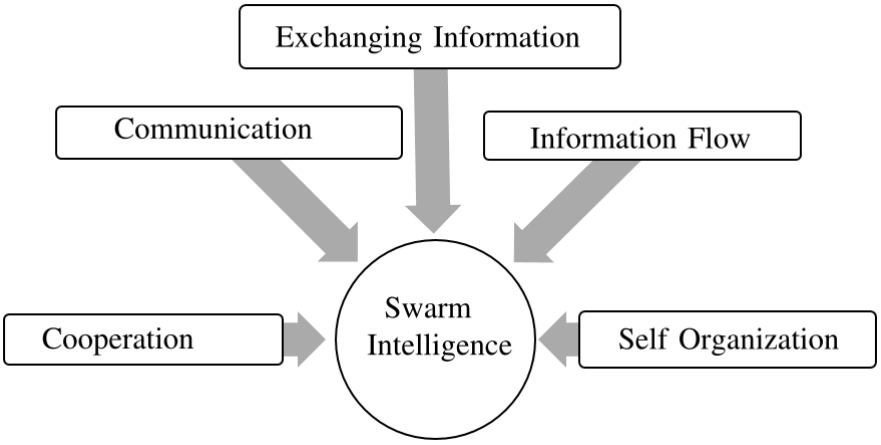


**Figure 4.1: Swarm intelligence**

In this work, several EC techniques have been used to solve the SSE and ED problems. Genetic algorithms (GAs) and several variants, such as breeder GA (BGA), fast navigating GA (FNGA), twin removal GA (TRGA), kite GA (KGA), and unified GA (UGA) have been implemented to solve the ED problem. Particle swarm optimization (PSO), differential evolution (DE), ant colony optimization (ACO), and harmony search (HS) have been used to tune the parameters of SVMs to solve the SSE problem. A modified particle swarm optimization (MPSO) technique has been proposed for tuning the SVR parameters for the purpose of tackling the online SSE problem. Furthermore, multi-objective particle swarm optimization (MOPSO) has been proposed for multi-objective FS in SSE.

GA variant based algorithms are presented in section 4.1, while PSO and MOPSO are discussed in section 4.2. Section 4.3 describes DE. Finally, ACO and HS are described in sections 4.4 and 4.5, respectively.

## 4.1 Variants of Genetic Algorithm

GA is a metaheuristic technique inspired by natural genetic populations to evolve a solution. The process of mutation, crossover, and selection in GA generate high-quality solutions to optimization and search problems. In this work, five different GA variants are implemented to solve the ED problem. Real-value encoding for this study has been used.

GA is an iterative algorithm that runs for a number of generations (epochs), which, in our implementation, is controlled by a predefined number of objective function evaluations ($Eval_{max}$), and terminates if the desirable solution is found. The five GAs that are studied in this work uses different genetic operators, including some recently introduced techniques, to produce new solutions for the next generation population. In following sections, breeder GA (BGA), fast navigating GA (FNGA), twin removal GA (TRGA), kite GA (KGA), and, unified GA (UGA) have been described [4].

### 4.1.1 Breeder Genetic Algorithm (BGA)

The breeder GA (BGA) [88, 89] includes three genetic operators during evolution: elitism, uniform crossover, and uniform single-point mutation. The procedure has been outlined in Algorithm 1. Elitism is an operator that segregates a subset of individuals from the current population at the beginning of a generation. This predefined proportion of chromosomes or individuals are relatively fitter than the others in the population, thus called elites. Elitism aims at the survival of these highly fitted individuals to guarantee non-decreasing GA performance over time. Both crossover and mutation can create or destroy the genetic material of a chromosome. Yet, elites are passed on to the next generation without any modification. Thus, elitism allows genetic material to be kept intact through evolution.

After elitism, roulette wheel algorithm has been used to select two parent chromosomes, $P_1$ and $P_2$, and uniform crossover has been applied on the parents to generate two offspring chromosomes, $O_1$ and $O_2$. In uniform crossover, the parents contribute to the offspring chromosomes in the gene (or variable) level, not at the segment level. A mixing ratio ($\alpha$) has been defined for each variable in the chromosome which is sampled from a uniform distribution within $[-0.1, +1.1]$. Then, $O_1 = \alpha P_1 + (1 - \alpha)P_2$ and $O_2 = \alpha P_2 + (1 - \alpha)P_1$. Finally, single-point uniform mutation has been applied to randomly selected candidates for mutation. Mutation is the process of randomly changing individuals of the current population to produce new individuals for the next generation. Mutation emulates the process of having random genetic diversity in nature. In

**Algorithm 1:** Procedure of BGA

```
Initialize population of individuals randomly;
while generation count ≤ Eval_max do
    Evaluate fitness of all chromosomes ;
    Perform elitism;
    Perform selection of parents for crossover;
    Perform uniform crossover (AmC);
    Perform uniform single-point mutation;
end
```

uniform mutation process, the value of one randomly chosen variable has been replaced, also referred as mutation point ($1 \leq mutation_p \leq d$, $d$ is the number of variables in a chromosome) in the mutation candidate with a uniform random value selected between the minimum and maximum variable values in that chromosome [4].

### 4.1.2 Fast Navigating Genetic Algorithm (FNGA)

The FNGA is a recently introduced GA variant [90] that uses elitism, a modified single-point crossover called AM-based crossover, and single-point mutation to introduce variation in the process of evolution, as outlined in Algorithm 2. At the beginning of the evolution, first a predefined number of fitter chromosomes has been segregated as elites, and these individuals have been passed to the next generation population without modification. Then modified single-point crossover operation has been applied. In the classical single-point crossover, one randomly selected locus is considered as the crossover point ($1 \leq corssover_p < (d-1)$) and the parts of the two parent chromosomes beyond $crossover_p$ are exchanged to produce two offspring chromosomes.

FNGA uses Associative Memory (AM)-based single-point crossover (AmC) [90] to enhance the constructive exploitation power of classical crossover. AM consists of two triangular memories that store the current best individual at all crossover points [90]. Unlike the classical crossover that blindly swaps the parts of parents, AmC produces two different offspring candidates for each offspring position: one taking the segment from another participating parent (classical crossover) and other using the segment available in AM. AmC takes feedback from the search space by evaluating the two potential offspring candidates and keeps the better one. Thus, the search for better solution variable in AmC operation is not limited to the other parent individual, but is rather extended

**Algorithm 2:** Procedure of FNGA

```
Initialize population of individuals randomly;
Initialize AM with the best available solution (chromosome);
while generation count ≤ Eval_max do
    Evaluate fitness of all chromosomes ;
    Perform global elitism;
    Perform selection of parents for crossover;
    Perform AM-based single-point crossover (AmC);
    Perform uniform single-point mutation;
end
```

to the current best solution that is stored in memory, and applied adaptively only if used. Moreover, the AM is updated if a better solution is found from the mating partner. Thus, AM can essentially contain the best-performing variables of a solution at different crossover point through consecutive generations. In addition to the AM-based crossover, FNGA uses single-point uniform mutation to ensure diversity in its population, as described in section 4.1.1. In our implementation of these different GAs, elites, as mutation candidates, have not been considered to always preserve the fitter solutions in the population [4].

### 4.1.3 Twin Removal Genetic Algorithm (TRGA)

Twin removal (TR) is an improved diversification operation, introduced in [91] for GA. The variant TRGA applies elitism, single-point crossover, uniform single-point mutation, and twin removal as genetic operators, as outlined in Algorithm 3. It has been discussed in [92] that GA tends to produce similar chromosomes called twins in the population as the generation proceeds. The growth of such correlated twins inevitably debilitates the impact of a mutation in producing new random solutions when the underlying landscape is complex. The TR operator can back up the reduced exploration power of mutation due to the similar chromosomes (twins) by introducing new random chromosomes in place of the similar chromosomesb [93].

In TRGA, elitism (see section 4.1.1) has been executed first and the classical single-point crossover on the selected parents has been executed to produce offsprings for the next generation. Thus, the parents contribute to the offspring chromosomes at the segment level where the parents exchange the subset of their variables at the $crossover_p$ (see section 4.1.2) to generate the offspring. The crossover operation is followed by the

**Algorithm 3:** Procedure of TRGA

```
Initialize population of individuals randomly;
while generation count ≤ Eval_max do
    Evaluate fitness of all chromosomes ;
    Perform elitism;
    Perform selection of parents for crossover;
    Perform single-point crossover;
    Perform uniform single-point mutation;
    Perform twin removal (TR);
end
```

single-point uniform mutation operation (see section 4.1.1). Finally, the TR operator has been exercised on the next generation population. The TR operation is controlled by the chromosome correlation factor (CCF) which defines the allowable similarity between chromosomes. The number of loci of the chromosome has been counted pair under comparison with identical values. For this application, CCF = 95% has been set and therefore if the similarity between two chromosomes is higher or equal than 95%, the chromosome has been replaced of relatively lower fitness with a new randomly generated chromosome [4].

### 4.1.4 Kite Genetic Algorithm (KGA)

The KGA [90] combines the enhanced exploitation capacity of AM-based crossover (see section 4.1.2) and improved diversification power of TR (see section 4.1.3). Thus, KGA has advantages over both FNGA and TRGA in having balanced exploitation and exploration in every generation. The flow of operations under procedure KGA has been shown in Algorithm 4. It has been laid down by the Schemata Theorem [55] that GA works by prioritizing and sustaining instances of the schema with above-average fitness. The AM-based crossover (AmC) ensures this by guiding the crossover towards better schema stored in the AM. Thus, it is more likely to exhibit similarity between chromosomes that can make the GA search static. TR plays a complementary role by reducing similarity and introducing new random solutions. Therefore, KGA employs elitism, AmC, single-point uniform mutation, and TR in one generation [4].

**Algorithm 4:** Procedure of KGA

```
Initialize population of individuals randomly;
Initialize AM with the best available solution (chromosome);
while generation count ≤ Eval_max do
    Evaluate fitness of all chromosomes ;
    Perform elitism;
    Perform selection of parents for crossover;
    Perform AM-based single-point crossover (AmC);
    Perform uniform single-point mutation;
    Perform twin removal (TR);
end
```

### 4.1.5 Unified Genetic Algorithm (UGA)

The UGA integrates a very recently proposed genetic operator in the literature, homologous gene replacement (hGR) [94] with the operators used in KGA. Thus, the unified GA (UGA) includes four genetic operators during evolution: elitism with hGR, AM-based single-point crossover, uniform single-point mutation and twin removal, summarized in Algorithm 5. The hGR works on the genes (or variables) of elites chromosomes and the elites, if possible. This operator is motivated to mimic the natural phenomena that the combination of good genes can form a fitter chromosome. The working principle of hGR operator is to identify the best gene (or variable) of each elite chromosome and replace the relatively weaker genes of the corresponding elite if this replacement improves that elite's fitness. Therefore, hGR involves the evaluation of relative fitness of the local variables to determine the best gene of an elite. To quantify the relative fitness of a gene in a chromosome (one variable in a solution), a common base value equal to 0.5 has been assigned to other variables to generalize the effect of other variables. The application of hGR is controlled adaptively by taking feedback from the search space to ensure the non-decreasing performance of GA. The benefits of hGR are first benchmarked in [95] where the authors used classical single-point crossover operation. However, in this GA variant, AM-based single-point crossover has been applied (see section 4.1.2) to utilize its improved intensification capacity. For diversification, both mutation (see section 4.1.1) and TR operatorhas been used (see section 4.1.3).

**Algorithm 5:** Procedure of UGA

```
Initialize population of individuals randomly;
Initialize AM with the best available solution (chromosome);
while generation count ≤ Eval_max do
    Evaluate fitness of all chromosomes ;
    Perform elitism with hGR;
    Perform selection of parents for crossover;
    Perform AM-based single-point crossover (AmC);
    Perform uniform single-point mutation;
    Perform twin removal (TR);
end
```

## 4.2 Particle Swarm Optimization

PSO is a population-based stochastic optimization technique in which every single solution is called a particle. It is inspired by the social behavior of bird flocks and fish schools [1]. Like other evolutionary methods, a population of random solutions is first created. Then, the algorithm looks for better solutions by updating the population in each iteration. At each iteration, $t$, the $i^{th}$ particle is aware of its own position, $P_i^t$, and velocity, $V_i^t$. It is also aware of the position of its personal best solution, $P_{Best,i}^t$, and of the best solution of the whole swarm, $G_{Best,i}^t$. For all particles, the fitness value is obtained by evaluating the function to be optimized at the particle position. The position of each particle in the next iteration is determined by its current position and velocity, and by the position of the personal and global best solutions. The stopping criteria are met when all particles reach an acceptable solution, or when a predefined number of iterations is reached. The velocity and position are updated as follows:

$$V_i^{t+1} = w^t V_i^t + c_1 r_1 (P_{Best}^t - P_i^t) + c_2 r_2 (G_{Best}^t - P_i^t) \tag{4.1}$$

$$P_i^{t+1} = P_i^t + V_i^t \tag{4.2}$$

$$w^{t+1} = w_{damp} w^t \tag{4.3}$$

where $c_1$ (cognition factor) and $c_2$ (social learning factor) are constants, $r_1$ and $r_2$ are random numbers in the range [0,1], and $w^t$ is the velocity weight at iteration $t$. Figure 4.2 shows the position updating process of a particle.

Velocity limits ($V_{max}$ and $V_{min}$) and position limits ($P_{max}$ and $P_{min}$) are set for each

variable. These limits can be set as follows:

$$V_{max} = 0.1(P_{max} - P_{min}), \quad V_{min} = -V_{max} \tag{4.4}$$

$$V_i^{t+1} = min(V_i^t, V_{max}), \quad V_i^{t+1} = max(V_i^t, V_{min}) \tag{4.5}$$

$$P_i^{t+1} = min(P_i^t, P_{max}), \quad P_i^{t+1} = max(P_i^t, P_{min}) \tag{4.6}$$

A bound limiting method can be used to return the particle back within the allowed range when a particle attempts to come out of the allowable limits. For instance, the velocity may be mirrored, as shown in Eq. 4.7.

$$if \;\; P_i^t \leq P_{min} \;\; or \;\; P_i^t \geq P_{max} \quad then \quad V_{i,o} = -V_{i,o} \tag{4.7}$$

In order to assist particle convergence, a constriction coefficient has been introduced [97]. This method aims at balancing the need for local and global search depending on the swarm conditions. In particular, based on the constriction coefficient method, the PSO parameters are defined as:

$$w = \chi, \quad c_1 = \chi\phi_1, \quad c_2 = \chi\phi_2 \tag{4.8}$$

where $\phi = \phi_1 + \phi_2 > 4$ and $\chi = \frac{2}{|\phi - 2 + \sqrt{\phi^2 - 4\phi}|}$.

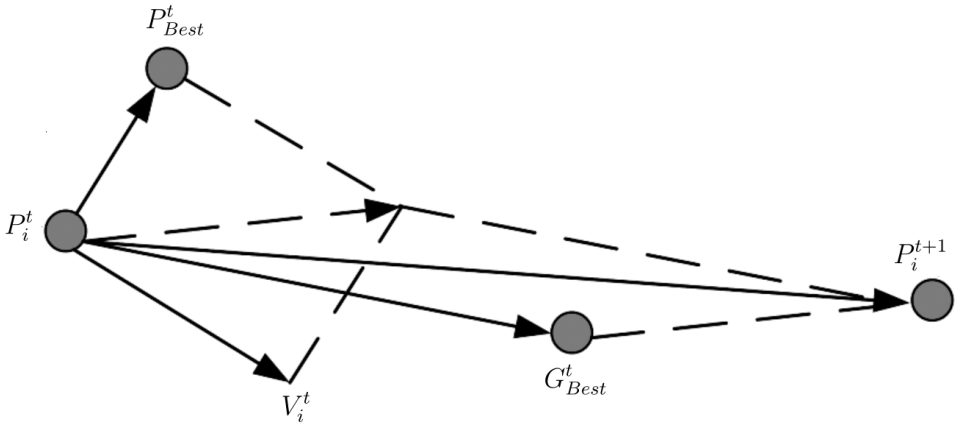


**Figure 4.2: Updating position of particle**

### 4.2.1 Multi-Objective Particle Swarm Optimization

PSO has a high-speed convergence and is suitable for multi-objective optimization. In [98], Pareto dominance is incorporated into PSO to extend its capability to handle multi-objective problems. A multi-objective algorithm is searching for a set of non-dominated solutions (Pareto optimal solutions). The set of all non-dominated solutions creates a trade-off surface or the Pareto front. A vector $x = \{x_1, x_2, \ldots, x_k\}$ is said to dominate $y = \{y_1, y_2, \ldots, y_k\}$ (Pareto dominance), if and only if $x$ is *partially less* than $y$, i.e., $\forall\, i \in \{1, 2, \ldots, k\},\ x_i \leq y_i \wedge \exists\, i \in \{1, 2, \ldots, k\} : x_i < y_i$.

MOPSO, which was proposed in [98], uses an external repository of particles from which some are picked as leader particles. The repository of particles is used by other particles to guide their own paths. The external repository, which consists of an archive controller and a grid, keeps a historical record of the non-dominated vectors found along the search process. The archive controller decides whether a certain solution should be added to the archive or not. The grid produces well-distributed Pareto fronts from the adaptive grid proposed in [99]. Combining the historical archive of non-dominated particles with the global best of the swarm guides the solutions to globally nond-ominated convergence.

MOPSO is initialized similarly to PSO by generating a random swarm and by updating the position and velocity of each particle. After evaluating each particle, a repository of non-dominated particles has to be created. $Rep_{p,i}$ and $Rep_{c,i}$ are the positions and the cost of particle $i$ in the repository. Then, the discovered objective space should be gridded by generating hypercubes in the search space, and the grid index of each member in the repository should be found [99]. The steps to create the grid are illustrated in Algorithm 6, where $\alpha$ is the inflation rate of the grid which helps to expand the grid and $N_{Obj}$ is the number of objectives (two in our problem).

The next step is selecting a leader $L$ from the repository. The steps to select a leader are outlined in Algorithm 7, where $\beta$ is the leader selection pressure, and $N_i$ is the number of particles in the occupied cell $i$. Velocity and position of each particle should be updated

**Algorithm 6:** Grid creation for MOPSO

Find minimum and maximum of $Rep_c$;
Find distance between minimum and maximum $d = Rep_{c,max} - Rep_{c,min}$;
Find min and max after inflation ($\alpha = 0.1$) ($Rep_{c,min} = Rep_{c,min} - d\alpha \quad Rep_{c,max} = Rep_{c,max} + d\alpha$) ;
for $k = 1 : nObj$;
$ck = linspace(Rep^k_{min}, Rep^k_{max}, nGrid + 1)$;
$Grid(k).LB = [-inf, ck]$;
$Grid(k).UB = [ck, +inf]$;
end

**Algorithm 7:** Leader selection for MOPSO

Find grid index of all the repository members;
Find occupied cells of the grid;
Find number of particles in the occupied cells;
Find selection probability from Boltzmann equation ($Ps_i = \frac{e^{-\beta N_i}}{\sum_j e^{-\beta N_j}}$);
Find selected cell index by roulette wheel selection to select the leader $L$;

by Eq. 4.9 and Eq. 4.10, respectively.

$$v_i^{t+1} = w^t v_i^t + c_1 r_1 (p^t_{i,Best} - p_i^t) + c_2 r_2 (L^t - p_i^t) \tag{4.9}$$

$$p_i^{t+1} = p_i^t + v_i^{t+1} \tag{4.10}$$

After updating the position, the cost function should be evaluated, and mutation could be applied to the population with probability mutation rate of $P_m = \frac{1}{\mu}(1 - \frac{t-1}{MaxIt-1})$. Mutation will lead to a new solution, $p^t_{New}$ ($\mu$ is the mutation rate), where $t$ is the iteration and $MaxIt$ is the maximum number of iterations. If the new solution dominates, the position and cost of each particle should be updated as:

$$p_i^{t+1} = \begin{cases} p_i^t & p_i^t \ dom \ p_{New} \\ p_{New} & p_{New} \ dom \ p_i^t \\ if \ \ rand < 0.5 \ \ p_i^t \ \ else \ \ p_{New} & otherwise \end{cases} \tag{4.11}$$

If the current position of the particle is better than the position contained in its memory, the particle's position is updated as $p_i^{t+1} = p^t_{i,Best}$ . Finally, add the non-dominated particles to the repository and update the grid and repository. The repository should be sorted and updated base on the geographical representations of particles in the grid. The stopping criteria are met when all particles reach an acceptable solution, or

when a predefined number of iterations is reached. Non-dominated solutions could be obtained from the historical record of $P_{Best}$.

## 4.3 Differential Evolution

DE is another heuristic optimization method which is suitable for high dimensional, non-linear, and non-differentiable continuous space function problems [82]. Similarly to other evolutionary techniques, DE consists of three operations, namely mutation, crossover, and selection. DE starts with a random initialization of the population vectors following a uniform distribution within the search space [83]. The idea behind DE is that crossover and mutation are used for generating trial vectors [12]. The mutated vector is generated by adding the weighted difference between two population vectors to a third vector. The newly generated vector is selected based on the crossover probability, $P_{CR} \in [0, 1]$, to ensure search diversity. Some of the newly generated vectors are used as offspring vectors for the next generation, while others remain unchanged [1].
The mutant vector is generated for each individual, $x_i$, as:

$$y_i = x_{r1} + \beta(x_{r2} - x_{r3}) \tag{4.12}$$

The position limits, $x_{max}$ and $x_{min}$, are applied for each individual such as:

$$x_i^{t+1} = min(x_i^t, x_{max}), x_i^{t+1} = max(x_i^t, x_{min}) \tag{4.13}$$

It is important that the three indices, $r_1$, $r_2$, $r_3$, are chosen to be different from each other, and they are selected randomly from within the population. Crossover is performed by combining the mutant vector, $y_i$, with the target vector, $x_i$, as follows:

$$z_i = \begin{cases} y_i & r_i \leq P_{CR} \quad or \quad i = i_0 \\ x_i & otherwise \end{cases} \tag{4.14}$$

$$P_{CR} = P_{CR,min} + \frac{(P_{CR,max} - P_{CR,min}).t}{MaxIter} \tag{4.15}$$

In Eqs. 4.14 and 4.15, $P_{CR} \in [0, 1]$ , $r_j$ is a random number between 0 and 1, $t$ indicates the iteration number, and $i_0 \in 1, 2, ..., d$, where $d$ represents the dimensionality. If the trial vector $z_i$ yields a better fitness than $x_i$, then $x_i$ is replaced by $z_i$, else $x_i$ is retained. The stopping criterion is met when all individuals achieve an appropriate minimum, or when the maximum number of iteration is reached.

## 4.4 Ant Colony Optimization for Continious Domain

The ants, which are almost blind insects, have the capability to cooperate in a colony to find the shortest route between the nest and a source of food. The first ant colony optimization (ACO) was proposed by Dorigo in the early 1990s [1]. ACO is able to deal with finding optimal combinations or permutations of variable problem components, and it was proposed to solve combinatorial optimization problems. In [84], ACO was extended to continuous domains. This new method, namely $ACO_R$, used a probability density function (PDF) instead of a discrete probability distribution.

$ACO_R$ is initialized with a uniform random sampling solution archive $S_l$ $(l = 1, 2, ..., n)$ where $n$ is the total population or archive size. The solutions in the archive are sorted based on their rank. Then, the weight vector, $w_l$ is computed as:

$$w_l = \frac{1}{\sqrt{2\pi}qn} \exp \frac{-(l-1)}{2(qn)^2} = N(l, 1, qn) \tag{4.16}$$

where $q$ is the intensification factor. The selection probability, $P_l$, is computed as:

$$P_l = \frac{w_l}{\sum_{l=1}^{n} w_l} \tag{4.17}$$

For the $l^{th}$ ant, the $l^{th}$ Gaussian function is computed in each iteration and the Gaussian kernel PDF is sampled using roulette wheel selection. The standard deviation is calculated based on average distance between solution $S_l$ and other solutions in the archive, as

follows:

$$\sigma_l = \zeta \sum_{r=1}^{n} \frac{|S_r - S_l|}{n-1} \tag{4.18}$$

where, $\zeta$, the deviation distance ratio, is a positive constant similar to pheromone evaporation ratio in ACO [84]. Finally, the new solution is generated by Eq. 4.19:

$$S_l^{new} = S_l + r_g \sigma_l \tag{4.19}$$

where $r_g$ is a random variable sampled from a Gaussian distribution. The whole process is repeated for each dimension until an acceptable minimum or a maximum number of iterations is reached. Pheromone update is accomplished by adding the best generated solutions to the archive and by eliminating the same number of worst solutions.

## 4.5 Harmony Search

HS is an optimization meta-heuristic algorithm inspired by the explicit principles of harmony improvisation, which is seeking a fantastic harmony (global optimum) [85]. Similarly to other meta-heuristic algorithms, a set of initial random vectors is chosen. In the case of HS, these vectors (harmonies), $\{H_1, H_2, ..., H_m\}$, are said to be filling the harmony memory (HM) [86]. A harmonic memory considering rate parameter, namely $HMCR \in [0, 1]$ is used to update the vectors in $HM$ in every iteration. More specifically, the $j^{th}$ element of a new vector, $H_{new}^j \in \{H_1^j, H_2^j, ..., H_m^j\}$ with probability $HMCR$, or a new random value is chosen with probability $1 - HMCR$. For improving the solutions and escaping from local optima, $H_{new}^j$ should be pitch-adjusted by the pitch adjusting rate (PAR). Specifically, if the random number in each iteration is smaller than PAR, then $H_{new}^j$ is replaced as follows.

$$H_{new}^j \longleftarrow H_{new}^j + \Delta \tag{4.20}$$

$$\alpha = 0.02(H_{max} - H_{min}) \tag{4.21}$$

$$\Delta = \alpha r \tag{4.22}$$

where $H_{max}$ and $H_{min}$ are, respectively, the decision variable upper and lower bounds, $\alpha$ is the fret width, which could be damp in each iteration, and $r$ is a random Gaussian number. New vectors should be added to HM if they provide better fitness values. In the end, the extra are truncated. The above steps repeat until a stopping criterion is reached [2].

# Chapter 5

# Proposed Techniques

In this chapter, the proposed AI techniques to solve the SSE and ED problems are presented. In section 5.1, the parameter selection method of multi-class SVM combined with EC methods for online SSE is described. In section 5.2, tuned support vector regression by modified PSO for online SSE is described. The FS method by MOPSO for SSE is explained in section 5.3. In section 5.4, a multi-classifier voter model for online SSE is proposed. Finally, in section 5.5, the description of implementing GA variant-based methods to solve the ED problem is presented.

## 5.1 Parameter Selection of Multi-Class SVM with EC Methods for Online SSE

In this section, we present a study of different EC optimization techniques for parameter selection of multi-class SVM to solve SSE online. In this work, SSE is viewed as a 2-class, a 3-class, or a 4-class classification problem. Commonly only 2 classes (secure and insecure) are considered in most works presented in the literature. Moreover, a number of optimization techniques are studied, including techniques which were not considered in previous works that used evolutionary optimization techniques to train SVMs.

More specifically, after employing a feature selection step, SVM with error-correcting output codes (ECOC) is used to address the multi-classification problem. The RBF kernel

is used to handle the nonlinearity. The performance of the SVM model strongly depends on the selection of the penalty factor, $C$, and the RBF kernel parameter, $\gamma$. Thus, it is essential to properly tune the SVM parameters. Four different EC methods (MPSO, DE, ACOr, and HS) are chosen to set the two SVM parameters. Each heuristic algorithm is initialized with a random population. The fitness value of each element (particle for MPSO, individual for DE, ant for ACOr, or harmony for HS) is evaluated for the whole population. It will be demonstrated that all methods provide almost identical classification accuracy, while HS operates faster than other methods. An important conclusion is that the level of accuracy for each technique depends on the number of classes, namely the number of security levels. The IEEE 39-bus system is used for implementing and validating the classifier performance. In the following subsection the implementation steps are provided.

### 5.1.1 Implementation Steps of Section 5.1

The IEEE 39-bus system was used to evaluate the proposed method. The information about this case study is presented in section 6.1.1. Since there were no real data available to check and compare the SSE performance, we had to generate the data used in the experimental studies. For generating the data off-line, different operating conditions were simulated by MatPower [87]. MatPower is a Matlab m-file package for solving power flow and optimal power flow problems. A traditional method, the Newton-Raphson load flow technique, was used to obtain the *true* SSI. The SSI was calculated using Eq. 2.5 and the classifier output for each classes are the labels presented on Tables 2.1, 2.2, and 2.3. The system load and generation were varied randomly in each case from 80% to 120% of their base values to prepare different scenarios. Around 60% of the total scenarios (1000 in this case) were associated with the $N-1$ contingency case (single line outage). Figure 5.1 shows the steps which were followed to generate the offline data. The input vectors are the voltage magnitude, $|V|_i$, and voltage angle, $\theta_i$, of each bus, $i$, as well as the complex power of each generator bus, $S_{Gi}$, the complex power load at each bus, $S_{Li}$, and the MVA power of each branch from bus $i$ to $j$, $S_{flow(ij)}$. By evaluating the SSI, each scenario is

**Table 5.1: Number of data in each class**

| 4 Classes | 3 Classes | 2 Classes |
|---|---|---|
| Class1: 261 | Class1: 261 | Class1: 568 |
| Class2: 307 | Class2: 466 | Class2: 432 |
| Class3: 159 | Class3: 273 | |
| Class4: 273 | | |
| Total Number of Data: 1000 | | |

marked to be in one of the possible classes.

The total number of cases for each class is presented in Table 5.1. About 80% of the data samples were randomly chosen for training, while 20% of the data were used for testing. Normalization was used for adjusting the values of each input variable in the range [0,1]. In particular, the maximum and minimum values of a particular variable were set to 1 and 0, respectively. Fifteen most important features were selected out of 157 total features by applying the SFS technique.

The implementation steps of parameter selection for SVM using several EC methods are shown in Figure 5.1. For tuning the SVM parameters, the algorithm starts with initializing a random population of members from training data. The parameters of each EC technique (MPSO, DE, ACOr, and HS) should be set at the beginning. Then, the fitness function of each member of the population should be evaluated. An SVM model is built for each member of the population, based on each member's parameters. For the experimental results presented in this work, 90% of the training samples were used for training, and 10% were used for validation. The fitness function can be calculated from the misclassified samples, as:

$$Missclass = \frac{No.\ of\ missclassified\ samples}{Total\ No.\ of\ samples}.100 \tag{5.1}$$

Then the population should be updated base on each EC algorithm. The algorithm can stop after a predefined maximum number of iteration. Otherwise the fitness function should be evaluated again. The simulation results are presented in section 6.2.1.

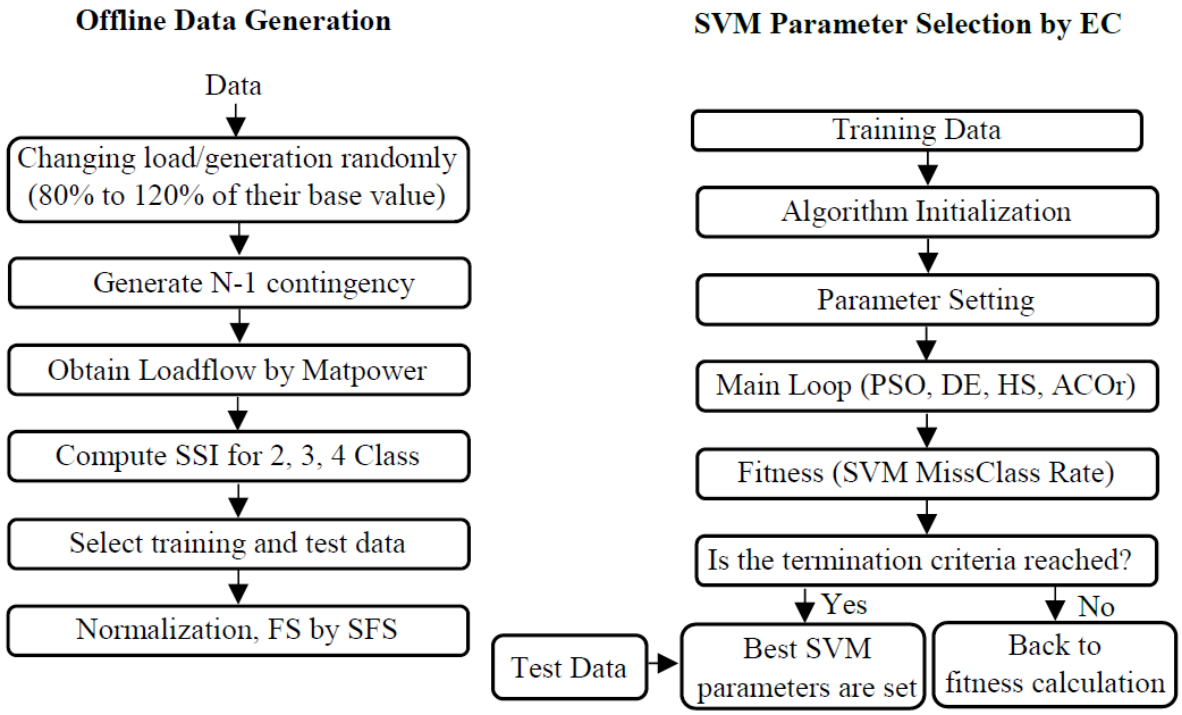


**Figure 5.1: Implementation procedure of section 5.1**

## 5.2 Tuned Support Vector Regression by Modified PSO for Online SSE

In this section, an improved technique based on SVR and PSO, namely the tuned support vector regression by modified particle swarm optimization (TSVR-MPSO) is proposed for online SSE. The performance of SVR heavily depends on the tuning of its parameters. More specifically, the three parameters which need to be properly tuned are the penalty parameter, $C$, the RBF kernel parameter, $\gamma$, and also the $\epsilon$ parameter (*see* section 3.2.2).

Recently, an approach was proposed for reliability prediction using SVR and PSO [24]. The main contribution in [24] is that different PSO particles are adapted using a different inertia weight based on an estimated global best. Inertia weight significantly affects the convergence and exploration and exploitation trade-off in PSO process [25]. In this work, an adaptation of the inertia weights is modified further, so that it is different for each one of the particle dimensions. In particular, in our proposed method, the PSO weight is updated for each particle based on the absolute distance between the global best and each particle's best position. The performance of TSVR-MPSO is compared with the method

in [24] and with another weight-adapting PSO technique, multilayer feed-forward neural network (MLFN) methods, and radial basis function network (RBFN) methods.

The objective of this work is to accurately predict the SSI for different contingency scenarios by training an SVR whose kernel parameters are optimized by PSO. The original SVR-PSO and the proposed method are presented in this section. The convergence behavior analysis in [24] concludes that the convergence speed of PSO is related to its inertia weight and prior knowledge about the global best can lead to better recognition degree. In [24], the ASPSO (Analytical selection PSO) inertia weight is updated as:

$$w_i^{t+1} = k^2 e^{-||P_{i,New}^t - X_{AS}||^2} \tag{5.2}$$

where $k$ is a constant, vector $X_{AS}$ is fixed based on the statistical properties of the training data [66], and $P_{i,New}$ is updated as:

$$P_{i,New}^t = \frac{c_1 r_1 p_{i,Best}^t + c_2 r_2 G_{Best}^t}{c_1 r_1 + c_2 r_2} \tag{5.3}$$

In [24], the distance in the exponent of Eq. 5.3 was the Mahalanobis with a covariance matrix $\Sigma$. However, in [24] it was assumed that $\Sigma$ is a unit matrix, which reduces the distance to the Euclidean, as shown in Eq. 5.3.

Based on the SVR-PSO theory, we propose TSVR-MPSO which associates the inertia weight with the global best position, $G_{best}^t$, instead of $X_{AS}$. The reasoning behind this modification is that both $X_{AS}$ and $G_{best}^t$ are estimates of the best solution. However, using $G_{best}^t$ eliminates the overhead needed to obtain $X_{AS}$, while $G_{best}^t$ is readily available in each iteration. Moreover, $G_{best}^t$ may be a better estimate than $X_{AS}$ as the iteration number increases. This modification leads to the following equation for updating the inertia weight as:

$$w_i^{t+1} = 0.95 e^{-||P_{i,Best}^t - G_{Best}^t||^2} \tag{5.4}$$

and we call the corresponding method TSVR-MPSO2. Additionally, since the different particle coordinates are associated with different parameter types, namely $C, \epsilon$, and $\gamma$, the above equation can be further modified so that each coordinate is updated according

to a different inertia weight. The TSVR-MPSO1 method updates the inertia weight as follows:

$$w_i^{t+1} = 0.95e^{-|P_{i,Best}^t - G_{Best}^t|^2} \tag{5.5}$$

Experimental results demonstrate that the TSVR-MPSO method provides a lower RMSE compared to other methods such as SVR-PSO, SVR-ASPSO, SVR-GS (SVR-grid search), RBFN, and MLFNN. The IEEE 14-bus and 118-bus test systems have been used to simulate our proposed technique.

### 5.2.1 Implementation Steps of Section 5.2

The IEEE 14-bus and 118-bus systems have been used to simulate proposed technique in section 5.2. The information about this case study is presented in section 6.1.1.

Since there were no real data available to check and compare SSE, we once again generated data using MatPower [87]. Different scenarios were generated by changing the load randomly from 90% to 110% of their base case. To evaluate the performance of the different techniques, the Newton-Raphson load flow technique was used to obtain the *true* SSI values. More specifically, the optimized Newton-Raphson load flow (NRLF) method was used for each scenario, to obtain the *true* bus voltages (magnitude and angle), and the line flows (net active and reactive power injected at each bus). Half of the scenarios were generated without an outage, while for the other half, an $N-1$ line outage contingency was considered. A total of 1000 and 4000 scenarios were generated for the IEEE 14-bus and 118-bus systems, respectively. For each experiment, 80% of the data vectors were used for training, and 20% were used for testing. The SSI was calculated using Eq. 2.5. The input vector consisted of parameters $P_G$ (real power of generator), $Q_G$ (reactive power of generator), $P_D$ (real power of load), $Q_D$ (reactive power of load), $|V|$ (magnitude voltage of PQ buses), $\delta$ (angle voltage of PQ buses) and the output is the SSI. Input vector elements were normalized in the range $[-1, 1]$ according to their type. For example, if $|V|_{max}$ and $|V|_{min}$ are, respectively, the maximum and minimum voltage magnitudes of all buses, the normalized voltage magnitude of the $k_{th}$ bus, $|V_n|_k$,

is calculated as follows:

$$|V_n|_k = \frac{2(|V_k| - |V|_{min})}{|V|_{max} - |V|_{min}} - 1 \tag{5.6}$$

The steps to offline data generation and the implementation process diagram are shown in Figure 5.2. The main steps of implementation are as follows:

1) Generate off-line data.

2) Initialize PSO. The $P^0 = \{P_1^0, P_2^0, ..., P_N^0\}$ are randomly set for $\varepsilon \in [0.0001, 0.0002], \gamma \in [0.1, 0.4]$ and $C \in [2000, 10000]$. $N$ is the swarm population.

3) Train SVR using each particle as SVR parameters.

4) Calculate fitness, which is the root mean square error (RMSE) between the SVR prediction and output of test data.

5) Find personal and global best positions, and update weight.

6) Repeat steps until $t$ reaches a set maximum number.

7) SVR prediction with optimal parameters obtained by MPSO.

The simulation results are presented in section 6.2.2.

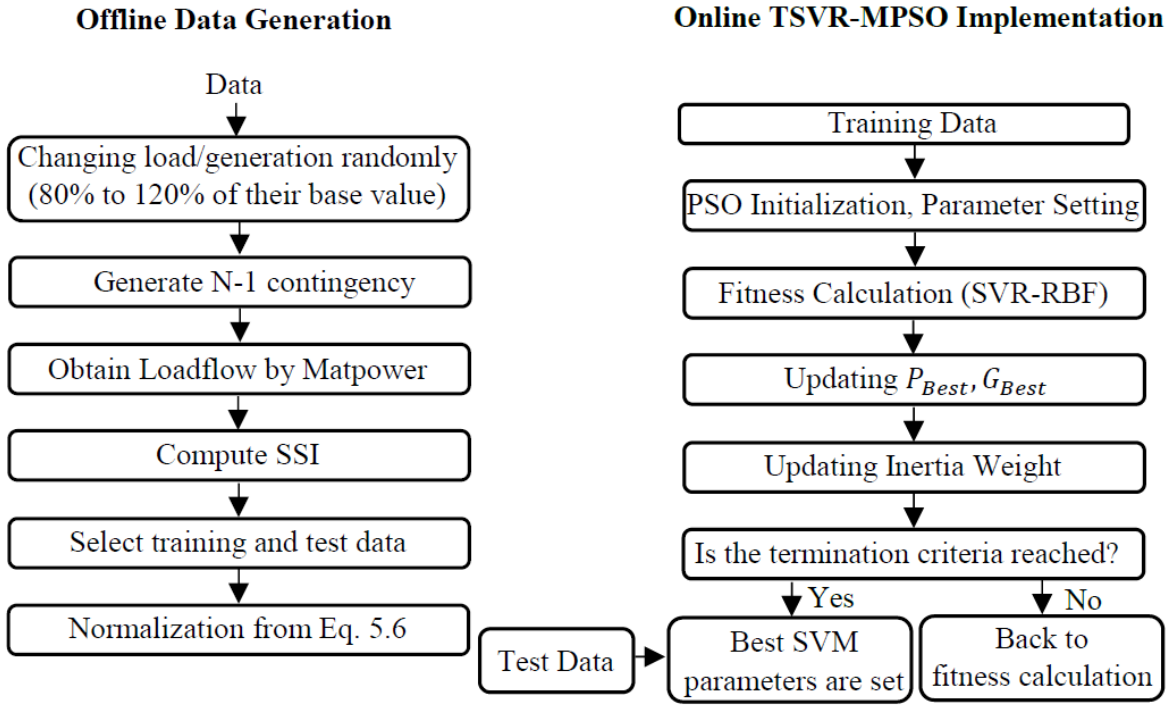


**Figure 5.2: Implementation steps of TSVR-MPSO**

## 5.3 Feature Selection by MOPSO for SSE

As mentioned in section 3.6, FS is an essential task for classification of data with many parameters. For the SSE problem, PMUs are unitized for collecting the necessary parameters from the power system. Reducing the number of features essentially implies a reduction of required PMUs. Using a smaller number of PMUs reduces the cost of the system. It also reduces the complexity of communication between the different units, resulting in a faster operation, which a requirement for online SSE. In previous literature, FS was also considered for solving the SSE problem. However, a predefined number of features had been set, and FS was solved as a single objective problem, mostly by the SFS method. However, FS is a multi-objective problem. The first objective is minimizing the classifier error, and the second objective is reducing the number of features. In this section, we use MOPSO (*see* section 4.2.1) for FS. The contribution of this work is using MOPSO for performing the FS for the SSE problem. The two objectives of the optimization problem are:

$$Fitness\ Function\ =\ Min \begin{cases} Z_1 = \frac{1}{N}\sum_{i=1}^{N}(t_i - Y_i)^2 \\ Z_2 = n_f \end{cases} \tag{5.7}$$

where $t$ is the target (the SSI) which was calculated by solving NR load flow equation (Eq. 2.5), $Y$ is the output from the random forest (RF) classifier model, and $N$ is the total number of samples. Fig. 5.3 demonstrates how the classifier error is determined for the first objective of the problem. In particular, a multi-classifier pattern is needed to model the system. The RF classifier (*see* section 3.3) is used for classification. The RF was chosen because it is simple, quick in terms of training, and no preparation regarding the input data is required. The vector $X = \{|V|, \theta, P_L, Q_L\}$ contains the features or inputs, namely the voltage magnitude and phase angle of PQ buses and the real and reactive power loads at all load buses. Then, $X_f$ are the selected features. The second objective is minimizing the number of features. In the following subsection, the implementation steps of the proposed technique are presented.

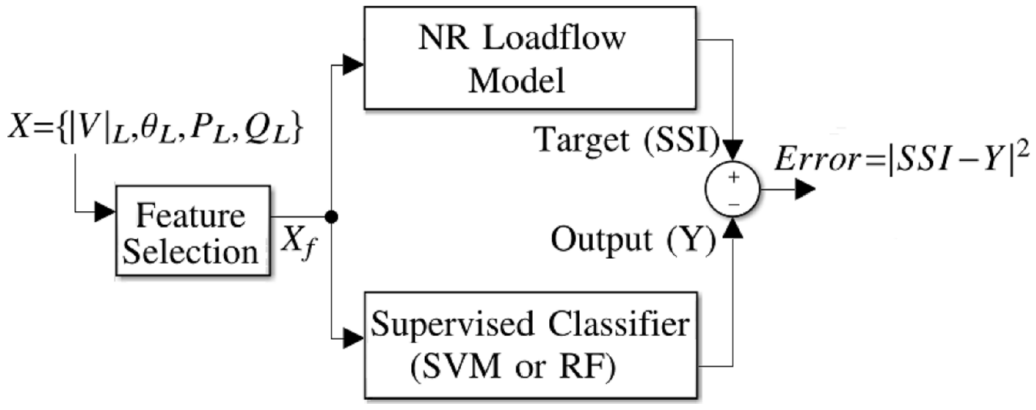


**Figure 5.3: Finding the classifier error**

### 5.3.1 Implementation Steps of Section 5.3

In this section, the implementation procedure of MOPSO (see section 4.2.1) for feature selection for SSE is illustrated. Fig. 5.4 demonstrates the implementation process. First, a set of random initial population should be created. Then, the MOPSO parameters should be set. These parameters are presented in Table 5.2. The swarm population and the repository size of each case study are presented in Table 5.3. Then, a repository and grid search should be generated from algorithm 6, and the fitness function from Eq. 5.7 should be evaluated. If the termination criterion is not satisfied, the leaders have to be selected from algorithm 7. The velocity and position of each particle have to be updated from Eqs. 4.9 and 4.10. Then, the fitness function should be evaluated again, and mutation has to be applied. Finally, the best particle should be updated from Eq. 4.11. The simulation results for this section are presented in section 6.2.3.

**Table 5.2: MOPSO parameters**

| Parameter | Vlaue |
|---|---|
| Personal learning coefficient | $c_1 = 1.4962$ |
| Global learning coefficient | $c_2 = 1.4962$ |
| Inflation Rate | $\alpha = 0.1$ |
| Leader selection pressure | $\beta = 2$ |
| Mutation rate | $\mu = 0.1$ |
| Number of grids per dimension | $nGrid = 5$ |

**Table 5.3: MOPSO population size**

| IEEE System | Swarm population | Repository size |
|---|---|---|
| 9-Bus | 10 | 5 |
| 14-Bus | 15 | 10 |
| 39-Bus | 20 | 15 |
| 57-Bus | 30 | 20 |
| 118-Bus | 50 | 30 |
| 300-Bus | 100 | 50 |

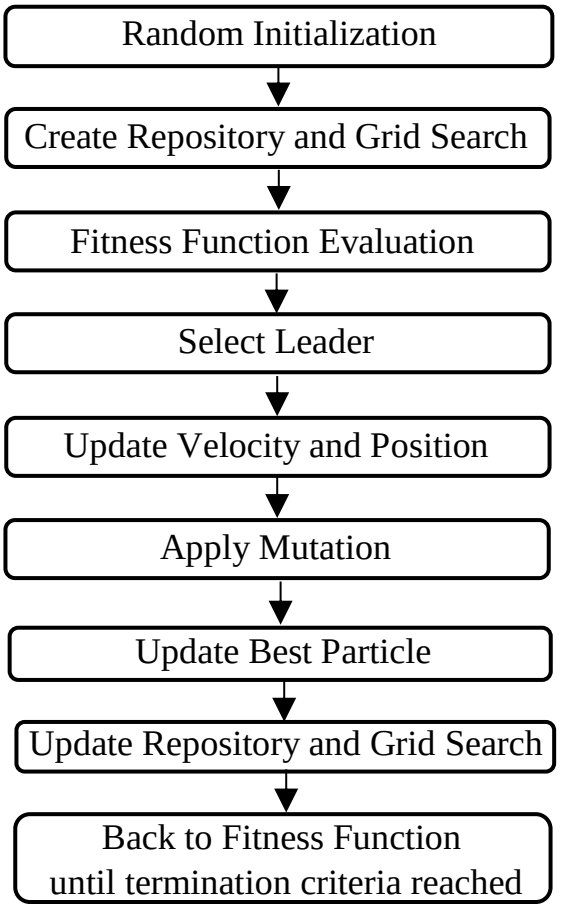


**Figure 5.4: MOPSO implementation for feature selection**

## 5.4 Multi-Classifier Voter Model for SSE

In this section, we propose a multi-classifier voter model for online SSE. We have concluded, by observing the performance of different nonlinear classifiers for the SSE problem, that different classifiers may exhibit superior performance depending on the particular

training and test data selection. Although the classification accuracy (CA) differences may not be always significant for small systems (such as the ones used in this work) they may be significant for larger systems. The proposed idea is based on the training of several classifiers and on allowing all models to work together for obtaining the SSI for the test data. After presenting the test data to each classifier, a simple voting scheme is applied at the classifier outputs, and the class label with the larger number of the votes is selected. Of course, it is apparent that a process which includes training several classifiers may be time-consuming. However, training the classifiers may be performed infrequently, and perhaps even only once. What is mainly important is for the testing process to be fast, which is actually the case for the proposed multi-classifier system. It should also be mentioned that although several classifier voting schemes have been used in the literature for other applications, we are not aware of one proposed for online SSE. In the following subsection, the implementation steps of the proposed technique are presented.

### 5.4.1 Implementation Steps of Section 5.4

In this work, nine multi-classifiers have been trained to perform the SSE. The first three classifiers are SVMs which use the polynomial kernel. Three other classifiers are also SVMs which use the RBF kernel. For these SVM three classifiers, the kernel parameters have been tuned by the MPSO technique. The implementation process of SVM-MPSO is shown in Fig. 5.5 and was explained in detail in section 5.2. The last three classifiers are random forests with an adaptive number of trees. After training each classifier, the nine models are saved and evaluated on the test data. A voting procedure ('mode') has been applied at the end, and the class label with a maximum number of votes is considered to be the final classification result. The implementation steps of the voting procedure are presented in Fig. 5.6, where $Y_1, Y_2, ..., Y_9$ are the outputs of the 9 classifiers, $N_{Test}$ is the total number of test data, $Y_{Test}$ is the output after voting, $SSI_i$ is the SSI of the $i^{th}$ sample, and $CA_i$ is a counter used for obtaining the number of correctly classified input vectors. The simulation results over several case studies are presented in section 6.2.4.

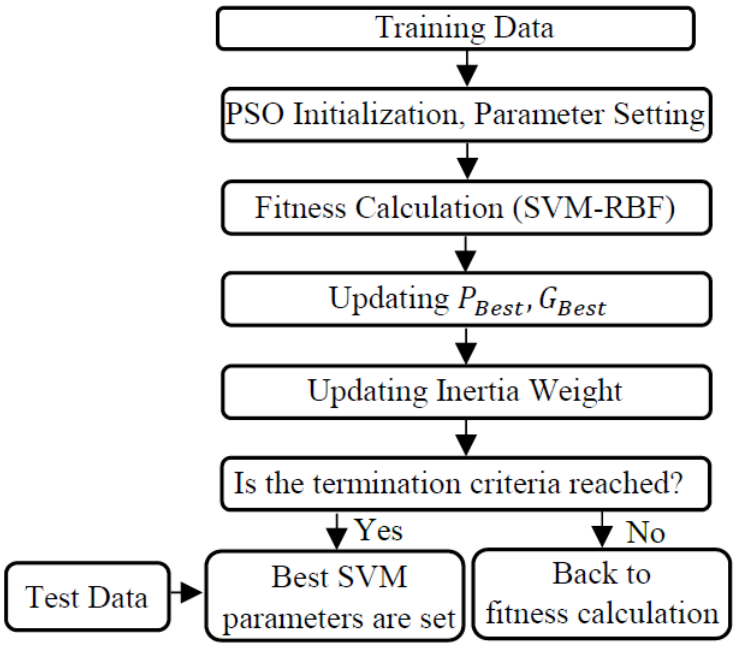


**Figure 5.5: Implementation steps of SVM-MPSO**

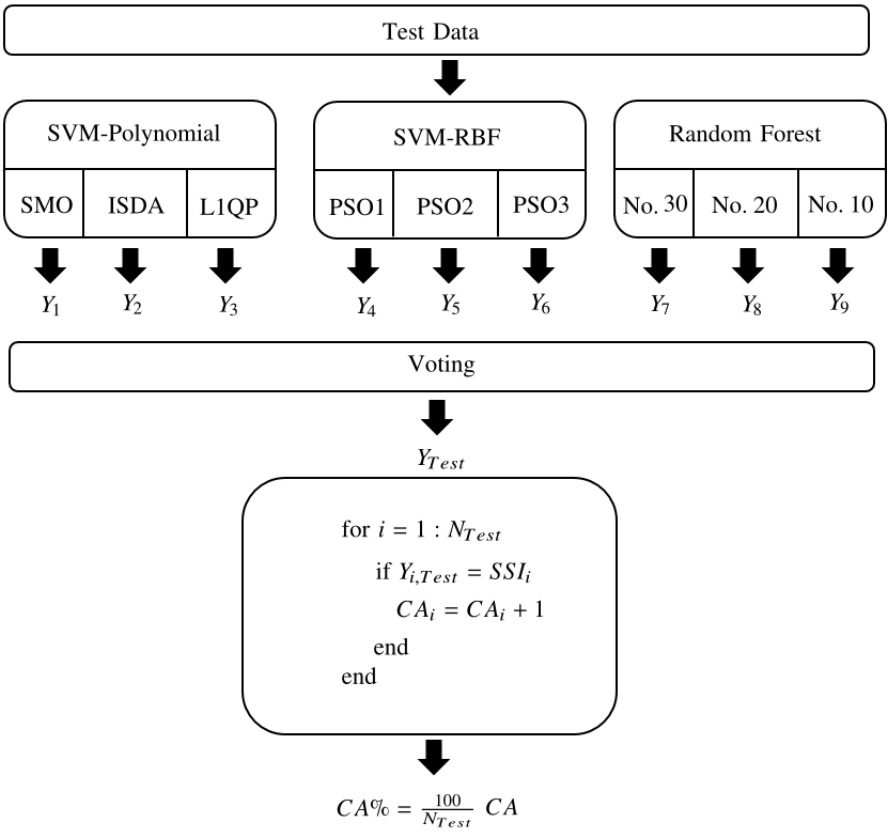


**Figure 5.6: Multi-classifier voter model**

## 5.5 Genetic Algorithm Variant based Effective Solutions for ED

ED is a nonlinear, nonconvex problem with several constraints. Traditionally, ED is solved numerically which is time-consuming and not effective. Recently, several EC algorithms have been used, proposed and tested to solve ED problems. In this work, advanced variants of GAs are used to solve the ED problem. Although ED problem was solved with GA algorithm before however, the contribution of this section is solving ED with several new GA variant based algorithms which have not been used to solve ED before.

The idea of GA was inspired by Darwin's theory of evolution and was first invented by John Holland. GA, in its implementation, starts with a set of individuals (initial population of candidate solutions) that are evolved over consecutive generations (epochs) through selection and variation to solve an optimization problem. In GA, individual problem-solutions, to which the values of the solution variables are encoded, are referred to as chromosomes. GA evolves through the natural adaptation process in which the fitter chromosomes tend to survive, breed, and propagate their genetic information to the future generations. The new proposed GA variants, with integrated advanced and innovative strategies, help produce competitive solutions. Though the GA variants were separately shown, a solution-suite can be easily formulated to have the combined benefits of the proposed GA variants.

In this work, BGA, FNGA, TRGA, KGA, and UGA (*see* section 4.1) have been tested on three IEEE benchmark systems, the 6-unit, 10-unit, and 15-unit systems (*see* section 6.1.2). It is also shown that for some case studies the KGA and TRGA outperform the results obtained by some recently proposed evolutionary and metaheuristic techniques, such as the modified social spider algorithm (MSSA) [50] and the backtracking search algorithm BSA [42]. In the following section the implementation steps to solve ED by GA variants is presented.

### 5.5.1 Implementation Steps of Section 5.5

For implementing the technique of section 5.5, the model of the system should be created first. The information about parameters of each case study is presented in section 6.1.2. The objective function and constraints should be created from Eqs. 2.6- 2.12. To handle the active power balance of the system (*see* Eq 2.9), a violation criterion has been set such as shown below:

$$violation = \begin{cases} 0 & P_{Total} > P_D + P_L \\ 1 - \frac{P_{Total} - P_L}{P_D} & otherwise \end{cases} \tag{5.8}$$

where, $P_{Total} = \sum_{j=1}^{n} P_j$. Therefor the objective function can set as:

$$Fitness\ Function = C_{Total}(1 + q.violation) \tag{5.9}$$

where $q$ was set to 100 in this work, and $C_{Total} = \sum_{j=1}^{n} F_j(P_j)$ (*see* Eq. 2.6). To use GA variant based algorithms for solving the ED problem, it is essential to normalize $P_j$, as:

$$P_{norm,j} = \hat{P}_{min} + (\hat{P}_{max} - \hat{P}_{min}) \tag{5.10}$$

where,

$$\hat{P}_{min} = max(P_j^{min}, P_j^0 - DR_j) \tag{5.11}$$

$$\hat{P}_{max} = min(P_j^{max}, P_j^0 + UR_j) \tag{5.12}$$

To handle the POZ, if $P_j$ is smaller than the average point of each interval, it should be set as the minimum value of the interval. Otherwise, it should be set as the maximum value of the interval. Then, the main loop of each GA algorithm should be initialized by setting the GA parameters, such as maximum number of iterations, population size, crossover percentage, number of parents and offsprings, mutation percentage, number of mutants, and the elitism rate. After generating a random population, the population should be sorted based on the fitness cost. The associative memory should be set later,

and the elitist selection, crossover, mutation, and twin removal should be coded based on the algorithms which are presented in section 4.1. The simulation results of each GA variant based algorithm is presented in section 6.2.5.

# Chapter 6

# Case Studies and Experimental Results

## 6.1 Case Studies

In this section, the different IEEE system case studies which have been used in this work are briefly described. In particular, section 6.1.1 describes the different IEEE bus systems which have been used for evaluating the performance of algorithms for the SSE problem. Section 6.1.2 describes the different IEEE unit systems which have been used for testing different algorithms for the ED problem.

### 6.1.1 Case Studies for SSE

For solving the SSE problem, the IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus systems have been used for the simulations. The total number of branches and generators of each system is presented in Table 6.1. The total number of PQ and PV buses of each case study is presented in Table 6.2. The single line diagrams of the IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus systems are shown in Figs. 6.1, 6.2, 6.3, 6.4, 6.5 and, 6.6, respectively. More information about bus data, generator data, and branch data of each case study is available at [87].

**Table 6.1: Number of branches and generators of case study**

| Case Study | No. Branches | No. Generators |
|---|---|---|
| IEEE 9 Bus | 9 | 3 |
| IEEE 14 Bus | 20 | 5 |
| IEEE 39 Bus | 46 | 11 |
| IEEE 57 Bus | 80 | 7 |
| IEEE 118 Bus | 186 | 54 |
| IEEE 300 Bus | 411 | 69 |

**Table 6.2: Total number of PQ and PV buses of each case study**

| Case Study | No. PQ buses | No. PV buses |
|---|---|---|
| IEEE 9 Bus | 6 | 2 |
| IEEE 14 Bus | 9 | 4 |
| IEEE 39 Bus | 29 | 9 |
| IEEE 57 Bus | 50 | 6 |
| IEEE 118 Bus | 64 | 53 |
| IEEE 300 Bus | 231 | 68 |

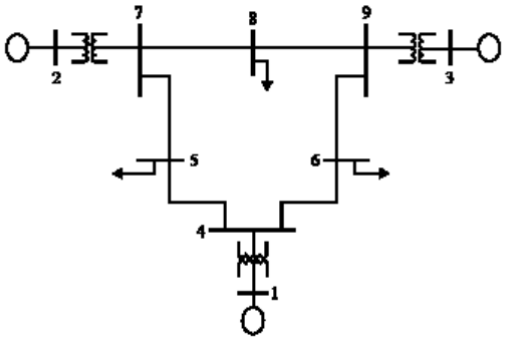


**Figure 6.1: IEEE 9-bus single line diagram [96]**

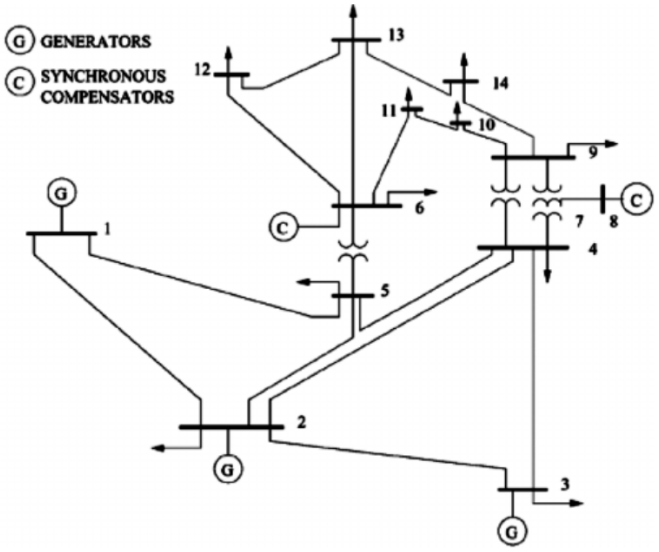


**Figure 6.2: IEEE 14-bus single line diagram[101]**

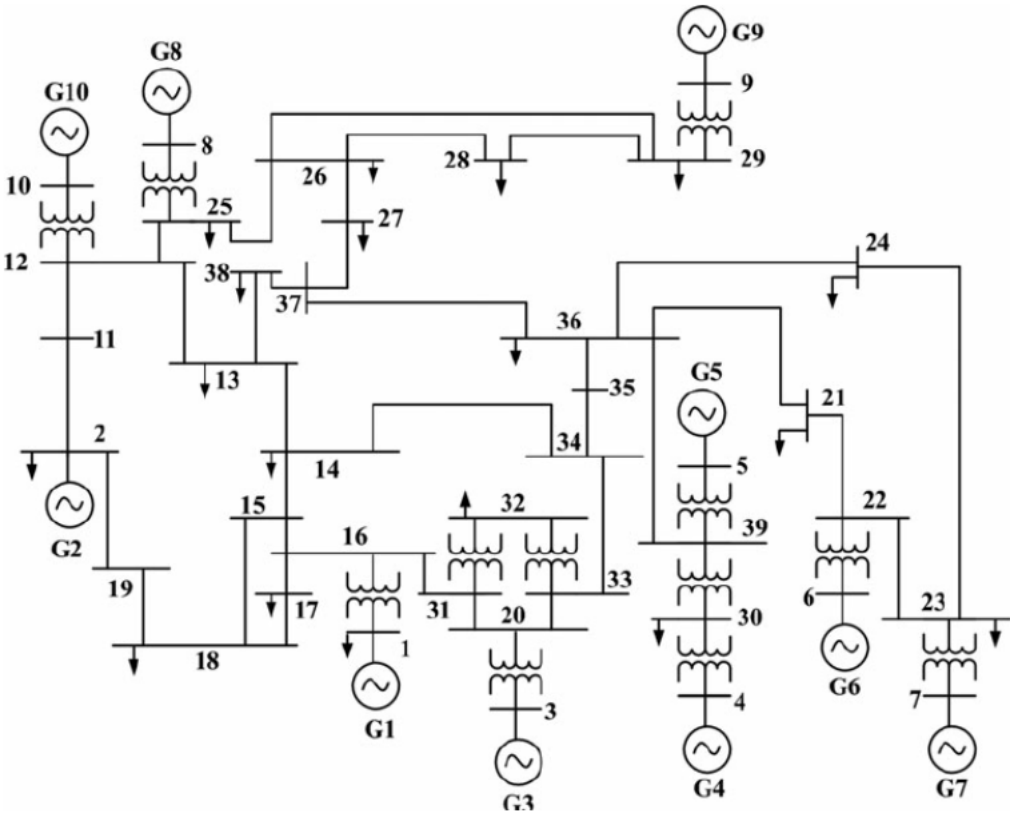


**Figure 6.3: IEEE 39-bus single line diagram [102]**

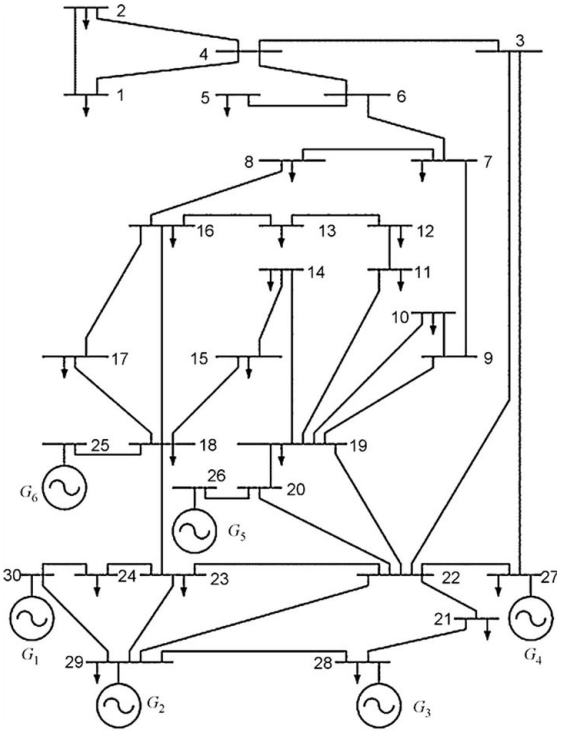


**Figure 6.4: IEEE 57-bus single line diagram [103]**

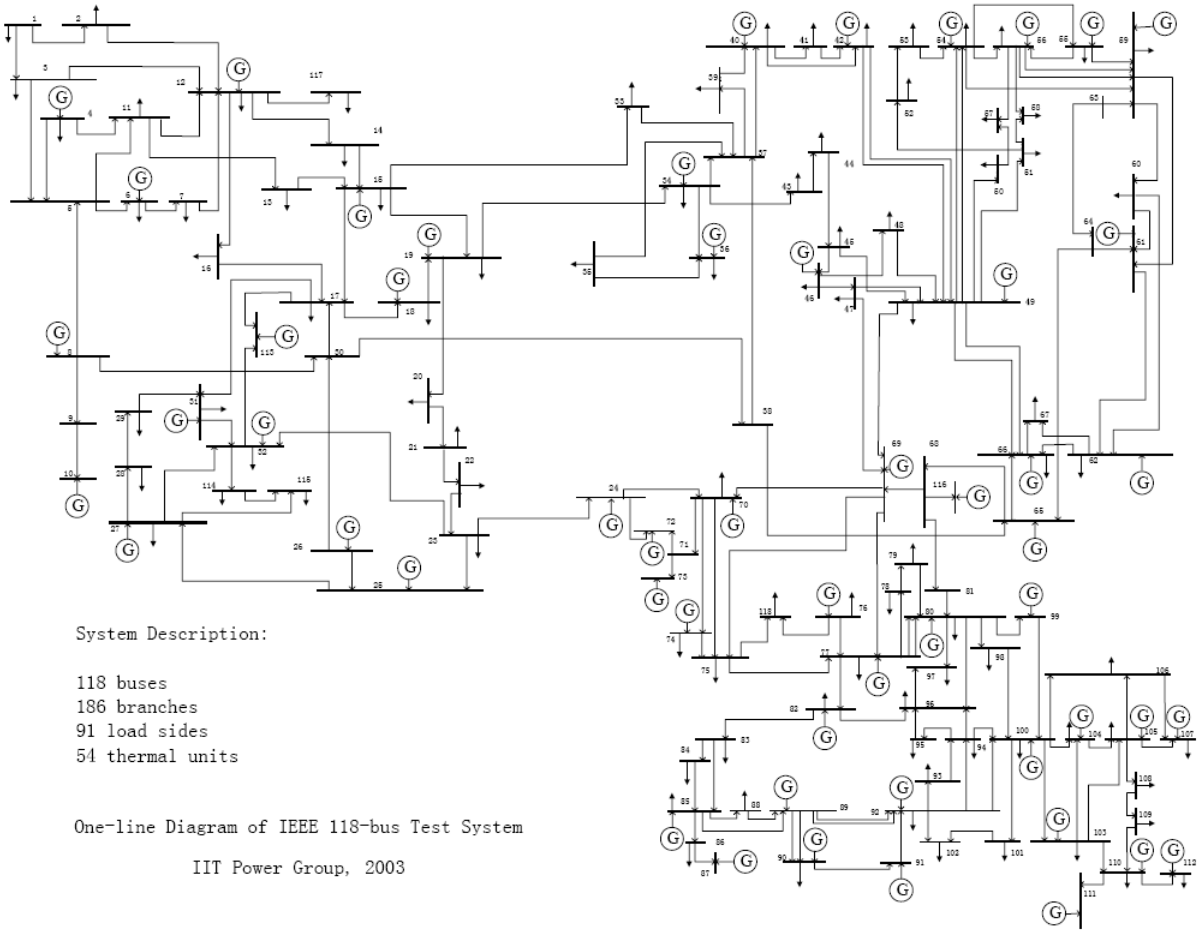


**Figure 6.5: IEEE 118-bus single line diagram [104]**

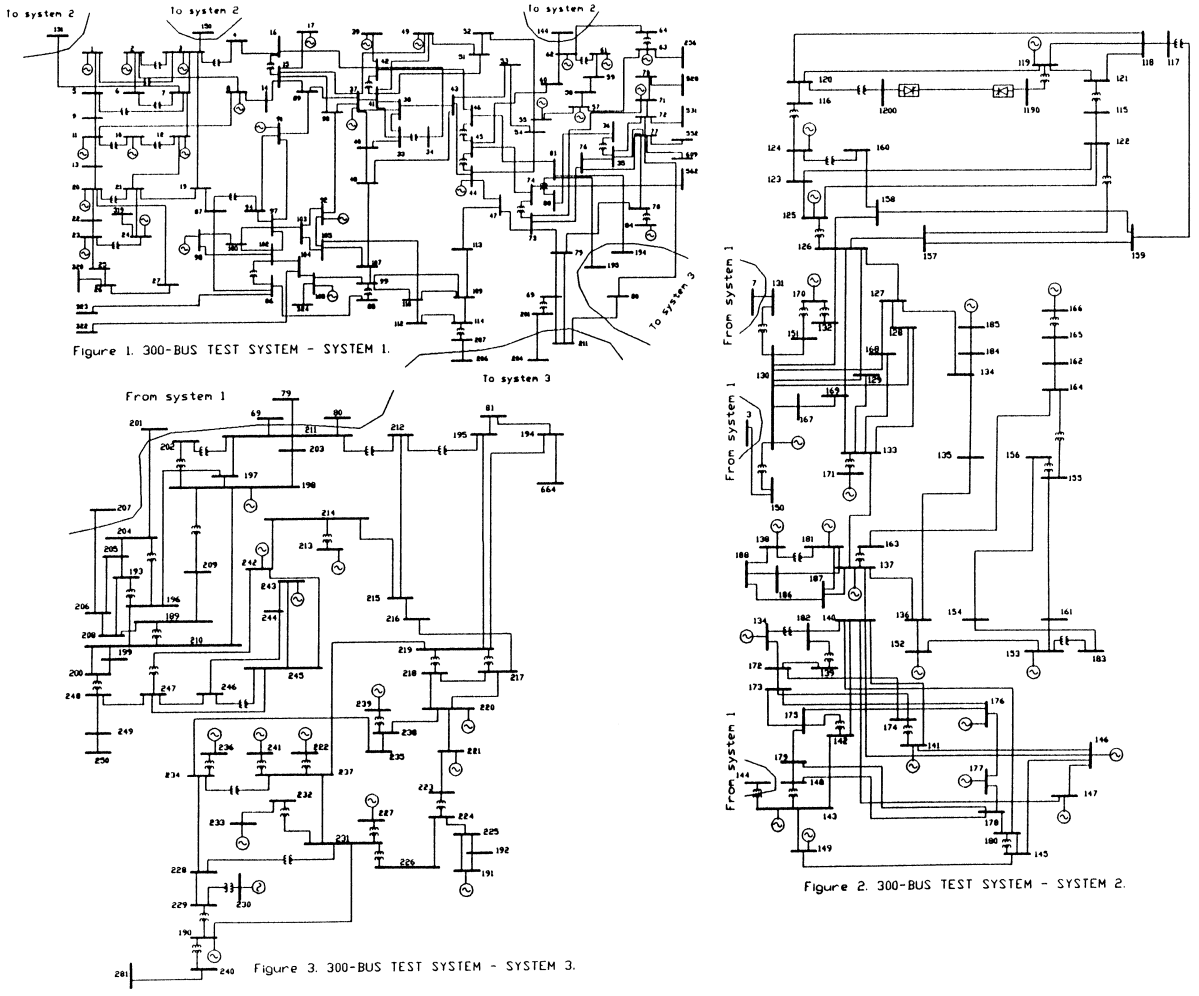


**Figure 6.6: IEEE 300-bus single line diagram [105]**

### 6.1.2 Case Studies for ED

In this section, the case studies which were used to solve ED are presented. Three popular benchmarks, namely the IEEE 6-unit, 15-unit, and 10-unit systems, were used to evaluate the performance of GA variants to solve ED. For the 6-unit and 15-unit systems, the total load demand is 1263 MW and 2630 MW, respectively. For these two cases, power balance, generator limits, ramp rate limits, and POZ (all described in Section 2.2) are used as constraints. For the 10-unit system, VPE and MFO are used as constraints. For this scenario, the load demand is 2700 MW. Generating unit capacity and coefficients of 6-unit, 10-units, and 15-units are presented in Tables 6.3, 6.4, and 6.6 respectively. the ramp rate limits and prohibited zones of 6-unit and 15-units are shown in Tables 6.5 and 6.7 respectively. The $B$ loss coefficient matrix of 6-unit and 15-unit are presented in Figs

6.7 and 6.8 respectively.

**Table 6.3: Generaing unit capacity and coefficients of 6-unit**

| **Unit** | $P_i^{min}$ | $P_j^{max}$ | $a_j$ | $b_j$ | $c_j$ |
|---|---|---|---|---|---|
| 1 | 100 | 500 | 240 | 7.0 | 0.0070 |
| 2 | 50 | 200 | 200 | 10.0 | 0.0095 |
| 3 | 80 | 300 | 220 | 8.5 | 0.0090 |
| 4 | 50 | 150 | 200 | 11.0 | 0.0090 |
| 5 | 50 | 200 | 220 | 10.5 | 0.0080 |
| 6 | 50 | 120 | 190 | 12.0 | 0.0075 |

$$B = 10^{-3}.\begin{bmatrix} 1.7 & 1.2 & 0.7 & -0.1 & -0.5 & -0.2 \\ 1.2 & 1.4 & 0.9 & 0.1 & -0.6 & -0.1 \\ 0.7 & 0.9 & 3.1 & 0 & -1 & -0.6 \\ -0.1 & 0.1 & 0 & 2.4 & -0.6 & -0.8 \\ -0.5 & -0.6 & -1 & -0.6 & 12.9 & -0.2 \\ -0.2 & -0.1 & -0.6 & -0.8 & -0.2 & 15 \end{bmatrix}$$

$$B_0 = 10^{-5}.\begin{bmatrix} -0.3908 & -0.1297 & 0.7047 & 0.0591 & 0.2161 & -0.6635 \end{bmatrix}$$

$$B_{00} = 0.56$$

**Figure 6.7: Loss coefficients for 6-unit**

$$B = 10^{-5}\begin{bmatrix} 1.4 & 1.2 & 0.7 & -0.1 & -0.3 & -0.1 & -0.1 & -0.1 & -0.3 & -0.5 & -0.3 & -0.2 & 0.4 & 0.3 & -0.1 \\ 1.2 & 1.5 & 1.3 & 0 & -0.5 & -0.2 & 0 & 0.1 & -0.2 & -0.4 & -0.4 & 0 & 0.4 & 1 & -0.2 \\ 0.7 & 1.3 & 7.6 & -0.1 & -1.3 & -0.9 & -0.1 & 0 & -0.8 & -1.2 & -1.7 & 0 & -2.6 & 11.1 & -2.8 \\ -0.1 & 0 & -0.1 & 3.4 & -0.7 & -0.4 & 1.1 & 5 & 2.9 & 3.2 & -1.1 & 0 & 0.1 & 0.1 & -2.6 \\ -0.3 & -0.5 & -1.3 & -0.7 & 9 & 1.4 & -0.3 & -1.2 & -1 & -1.3 & 0.7 & -0.2 & -0.2 & -2.4 & -0.3 \\ -0.1 & -0.2 & -0.9 & -0.4 & 1.4 & 1.6 & 0 & -0.6 & -0.5 & -0.8 & 1.1 & -0.1 & -0.2 & -1.7 & 0.3 \\ -0.1 & 0 & -0.1 & 1.1 & -0.3 & 0 & 1.5 & 1.7 & 1.5 & 0.9 & -0.5 & 0.7 & 0 & -0.2 & -0.8 \\ -0.1 & 0.1 & 0 & 5 & -1.2 & -0.6 & 1.7 & 16.8 & 8.2 & 7.9 & -2.3 & -3.6 & 0.1 & 0.5 & -7.8 \\ -0.3 & -0.2 & -0.8 & 2.9 & -1 & -0.5 & 1.5 & 8.2 & 12.9 & 11.6 & -2.1 & -2.5 & 0.7 & -1.2 & -7.2 \\ -0.5 & -0.4 & -1.2 & 3.2 & -1.3 & -0.8 & 0.9 & 7.9 & 11.6 & 20 & -2.7 & -3.4 & 0.9 & -1.1 & -8.8 \\ -0.3 & -0.4 & -1.7 & -1.1 & 0.7 & 1.1 & -0.5 & -2.3 & -2.1 & -2.7 & 14 & 0.1 & 0.4 & -3.8 & 16.8 \\ -0.2 & 0 & 0 & 0 & -0.2 & -0.1 & 0.7 & -3.6 & -2.5 & -3.4 & 0.1 & 5.4 & -0.1 & -0.4 & 2.8 \\ 0.4 & 0.4 & -2.6 & 0.1 & -0.2 & -0.2 & 0 & 0.1 & 0.7 & 0.9 & 0.4 & -0.1 & 10.3 & -10.1 & 2.8 \\ 0.3 & 1 & 11.1 & 0.1 & -2.4 & -1.7 & -0.2 & 0.5 & -1.2 & -1.1 & -3.8 & -0.4 & -10.1 & 57.8 & -9.4 \\ -0.1 & -0.2 & -2.8 & -2.6 & -0.3 & 0.3 & -0.8 & -7.8 & -7.2 & -8.8 & 16.8 & 2.8 & 2.8 & -9.4 & 128.3 \end{bmatrix}$$

$$B_0 = 10^{-4}\begin{bmatrix} -1 & -2 & 28 & -1 & 1 & -3 & -2 & -2 & 6 & 39 & -17 & 0 & -32 & 67 & -64 \end{bmatrix}$$

$$B_{00} = 0.55$$

**Figure 6.8: Loss coefficients for 15-unit**

## 6.2 Simulation Results

In this section, the simulations results of all proposed techniques in chapter 5 are presented. The experiments were performed using MATLAB 8.6.0 (R2015b) running Windows 10 on an i5-42000U CPU 2.29 GHz.

**Table 6.4: Generaing unit capacity and coefficients of 15-unit**

| **Unit** | $P_j^{min}$ | $P_j^{max}$ | $a_j$ | $b_j$ | $c_j$ |
|---|---|---|---|---|---|
| 1 | 150 | 455 | 671 | 10.1 | 0.000299 |
| 2 | 150 | 455 | 574 | 10.2 | 0.000183 |
| 3 | 20 | 130 | 374 | 8.8 | 0.001126 |
| 4 | 20 | 130 | 374 | 8.8 | 0.001126 |
| 5 | 150 | 470 | 461 | 10.4 | 0.000205 |
| 6 | 135 | 460 | 630 | 10.1 | 0.000301 |
| 7 | 135 | 465 | 548 | 9.8 | 0.000364 |
| 8 | 60 | 300 | 227 | 11.2 | 0.000338 |
| 9 | 25 | 162 | 173 | 11.2 | 0.000338 |
| 10 | 25 | 160 | 175 | 10.7 | 0.001203 |
| 11 | 20 | 80 | 186 | 10.2 | 0.003586 |
| 12 | 20 | 80 | 186 | 10.2 | 0.000807 |
| 13 | 25 | 85 | 225 | 13.1 | 0.000371 |
| 14 | 15 | 55 | 309 | 12.1 | 0.001929 |
| 15 | 15 | 55 | 323 | 12.4 | 0.004447 |

**Table 6.5: Ramp rate limits and prohibited zones of 6-unit**

| **Unit** | $P_j^0$ | $UR_j(MW/h)$ | $DR_j(MW/h)$ | **Prohibited zones** $(MW)$ |
|---|---|---|---|---|
| 1 | 440 | 80 | 120 | [210 240][350 380] |
| 2 | 170 | 50 | 90 | [90 110][140 160] |
| 3 | 200 | 65 | 100 | [150 170][210 240] |
| 4 | 150 | 50 | 90 | [80 90][110 120] |
| 5 | 190 | 50 | 90 | [90 110][140 150] |
| 6 | 110 | 50 | 90 | [75 85][100 105] |

**Table 6.6: System coefficients for10-unit test system with VPE and MFO**

| Unit | fuel | $P_j^{min}$ | $P_j^{max}$ | $a_j$ | $b_j$ | $c_j$ | $e_j$ | $f_j$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 100 | 250 | 26.97 | -0.3975 | 0.002176 | 0.02697 | -3.975 |
| 1 | 2 | 100 | 250 | 21.13 | -0.3059 | 0.001861 | 0.02113 | -3.059 |
| 2 | 1 | 50 | 230 | 118.4 | -1.269 | 0.004194 | 0.1184 | -12.69 |
| 2 | 2 | 50 | 230 | 1.865 | -0.0399 | 0.001138 | 0.00187 | -0.3988 |
| 2 | 3 | 50 | 230 | 13.65 | -0.198 | 0.00162 | 0.01365 | -1.98 |
| 3 | 1 | 200 | 500 | 39.79 | -0.3116 | 0.001457 | 0.03979 | -3.116 |
| 3 | 2 | 200 | 500 | -59.14 | 0.4864 | 1.18E-05 | -0.05914 | 4.864 |
| 3 | 3 | 200 | 500 | -2.876 | 0.0339 | 0.000804 | -0.00288 | 0.3389 |
| 4 | 1 | 99 | 265 | 1.983 | -0.0311 | 0.001049 | 0.00198 | -0.3114 |
| 4 | 2 | 99 | 265 | 52.85 | -0.6348 | 0.002758 | 0.05285 | -6.348 |
| 4 | 3 | 99 | 265 | 266.8 | -2.338 | 0.005935 | 0.2668 | -23.38 |
| 5 | 1 | 190 | 490 | 13.92 | -0.0873 | 0.001066 | 0.01392 | -0.8733 |
| 5 | 2 | 190 | 490 | 99.76 | -0.5206 | 0.001597 | 0.09976 | -5.206 |
| 5 | 3 | 190 | 490 | -53.99 | 0.4462 | 0.00015 | -0.05399 | 4.462 |
| 6 | 1 | 85 | 265 | 52.15 | -0.6348 | 0.002758 | 0.05285 | -6.348 |
| 6 | 2 | 85 | 265 | 1.983 | -0.0311 | 0.001049 | 0.00198 | -0.3114 |
| 6 | 3 | 85 | 265 | 266.6 | -2.338 | 0.005935 | 0.2668 | -23.38 |
| 7 | 1 | 200 | 500 | 18.93 | -0.1325 | 0.001107 | 0.01893 | -1.325 |
| 7 | 2 | 200 | 500 | 43.77 | -0.2267 | 0.001165 | 0.04377 | -2.267 |
| 7 | 3 | 200 | 500 | 43.35 | 0.3559 | 0.000245 | -0.04335 | 3.559 |
| 8 | 1 | 99 | 265 | 1.983 | -0.0311 | 0.001049 | 0.00198 | -0.3114 |
| 8 | 2 | 99 | 265 | 52.85 | -0.6348 | 0.002758 | 0.05285 | -6.348 |
| 8 | 3 | 99 | 265 | 266.8 | -2.338 | 0.005935 | 0.2668 | -23.38 |
| 9 | 1 | 130 | 440 | 88.53 | -0.5675 | 0.001554 | 0.08853 | -5.675 |
| 9 | 2 | 130 | 440 | 15.32 | -0.0451 | 0.007033 | 0.01423 | -0.1817 |
| 9 | 3 | 130 | 440 | 14.23 | -0.0182 | 0.000612 | 0.01423 | -0.1817 |
| 10 | 1 | 200 | 490 | 13.97 | -0.0994 | 0.001102 | 0.01397 | -0.9938 |
| 10 | 2 | 200 | 490 | -61.13 | 0.5084 | 4.16E-05 | -0.06113 | 5.084 |
| 10 | 3 | 200 | 490 | 46.71 | -0.2024 | 0.001137 | 0.04671 | -2.024 |

### 6.2.1 Simulation Results of Parameter Selection of Multi-Class SVM with EC Methods for Online SSE

As it is mentioned in section 5.1, the performance of the SVM model is strongly dependent on the selection of the penalty factor, $C$, and the RBF kernel parameter $\gamma$. Four different heuristic optimization methods (MPSO, DE, ACOR, and HS) are chosen to set two SVM parameters. Each heuristic algorithm is initialized with a random population. The fitness value of each element (particle for MPSO, individual for DE, ant for ACOR, or harmony for HS) is evaluated for the whole population. A population size of 20 and a

**Table 6.7: Ramp rate limits and prohibited zones of 15-unit**

| **Unit** | $P_j^0$ | $UR_j(MW/h)$ | $DR_j(MW/h)$ | **Prohibited zones** $(MW)$ |
|---|---|---|---|---|
| 1 | 80 | 120 | 400 | - |
| 2 | 80 | 120 | 300 | [185 225][305 335][420 450] |
| 3 | 130 | 130 | 105 | - |
| 4 | 130 | 130 | 100 | - |
| 5 | 80 | 120 | 90 | [180 200][305 335][390 420] |
| 6 | 80 | 120 | 400 | [230 255][365 395][430 455] |
| 7 | 80 | 120 | 350 | - |
| 8 | 65 | 100 | 95 | - |
| 9 | 60 | 100 | 105 | - |
| 10 | 60 | 100 | 110 | - |
| 11 | 80 | 80 | 60 | - |
| 12 | 80 | 80 | 40 | [30 40][55 65] |
| 13 | 80 | 80 | 30 | - |
| 14 | 55 | 55 | 20 | - |
| 15 | 55 | 55 | 20 | - |

search space limit of $\gamma \in [2^{-4}, 4^4]$ and $C \in [2^5, 2^{15}]$ are used for all four heuristic methods. Table 6.8 shows the values of the parameters associated with each heuristic method. A total of 100 trials were performed for each method. The correct classification rate (CCR) for each individual class, as well as the average CCR considering all classes are presented in 6.9, 6.10, and 6.11. The total training time for all classes of each method is provided in 6.12.

MPSO provided the best CCR (92.68%) for the 4-class problem with $C = 14536.97$ and $\gamma = 3.43$, while ACOr provided the highest CCR (95.07%) for the 3-class problem with $C = 14143.62$ and $\gamma = 3.61$. Then, HS provided a slightly higher CCR (97.42%) compared to the other methods for the 2-class problem with $C = 11884.56$ and $\gamma = 4.27$. In all experiments, HS exhibited the fastest training speed. The training CCR for all methods was 100%.

**Table 6.8: Parameters set for each optimization method**

| Method | Parameters |
|---|---|
| **PSO** | $C_1 = C_2 = 1.4192,\ w = 0.8298,\ w_{damp} = 0.98$ |
| **DE** | $\beta = 0.05, P_{CR_{min}} = 0.1, P_{CR_{max}} = 0.9$ |
| **ACOr** | $q = 0.5,\ \zeta = 1$ |
| **HS** | $HM_{size} = 10, H_{new,size} = 10, HMCR = 0.5, PAR = 0.24$ |

**Table 6.9: Results for 4 classification problem**

| Testing Phase CCR% | | | | |
|---|---|---|---|---|
| | **SVM-MPSO** | **SVM-DE** | **SVM-ACOr** | **SVM-HS** |
| **Class1** | 92.42% | 91.44% | 90.68% | 90.08% |
| **Class2** | 89.75% | 90.14% | 89.34% | 87.86% |
| **Class3** | 78.52% | 77.78% | 73.76% | 72.20% |
| **Class4** | 94.12% | 95.51% | 95.26% | 94.53% |
| **Total** | **92.68**% | 91.89% | 91.65% | 91.65% |

**Table 6.10: Results for 3 classification problem**

| Testing Phase CCR% | | | | |
|---|---|---|---|---|
| | **SVM-MPSO** | **SVM-DE** | **SVM-ACOr** | **SVM-HS** |
| **Class1** | 92.93% | 92.42% | 93.31% | 92.12% |
| **Class2** | 92.97% | 92.74% | 93.32% | 92.67% |
| **Class3** | 94.91% | 96.65% | 94.58% | 91.15% |
| **Total** | 94.63% | 94.48% | **95.07**% | 94.42% |

**Table 6.11: Results for 2 classification problem**

| Testing Phase CCR% | | | | |
|---|---|---|---|---|
| **CCR%** | **SVM-MPSO** | **SVM-DE** | **SVM-ACOr** | **SVM-HS** |
| **Class1** | 96.68% | 97.06% | 96.03% | 95.68% |
| **Class2** | 94.49% | 94.83% | 94.40% | 94.83% |
| **Total** | 97.31% | 97.12% | 97.31% | **97.42**% |

**Table 6.12: Training Time (sec) of multi-class SVM with EC methods for SSE**

| Training Time (sec) | | | | |
|---|---|---|---|---|
| **Number of Classes** | **SVM-MPSO** | **SVM-DE** | **SVM-ACOr** | **SVM-HS** |
| **Four** | 3.9781 | 3.7962 | 2.2009 | **2.0091** |
| **Three** | 2.9766 | 3.2723 | 1.8637 | **1.6313** |
| **Two** | 4.2203 | 4.0573 | 2.4522 | **2.2309** |

### 6.2.2 Simulations Results of Tuned Support Vector Regression by Modified PSO for Online SSE

In this section, the simulation results of proposed technique in section 5.2 is presented. The prediction accuracy of different methods for SSE is assessed for the IEEE 14-bus and 118-bus systems. In particular, the SSI obtained by these methods is compared with the SSI obtained by the NRLF method, assuming that NRLF is accurate. The RMSE between predicted and true SSI values for a set of test data is used for comparison. The methods tested are:

- SVR-PSO: Inertia weight decreasing as iterations increase,
- TSVR-MPSO1 and TSVR-MPSO2: Inertia weight updated for each particle based on, respectively, the absolute and Euclidean distance between personal and global best,
- SVR-ASPSO: Inertia weight updated as in Eq. (5.2),
- SVR-GS: Basic grid search method which adopts v-fold cross-validation for tuning the SVR,
- RBFN: For three different set of parameters, and
- MLFFN: For three different set of parameters.

Specific information about the different parameters of all methods mentioned above is presented in Tables 6.13, 6.17, and 6.19 present the prediction accuracy results for IEEE 14-bus and 118-bus for 20 different test experiments.

As shown in Tables 6.17 and 6.19, the proposed TSVR-MPSO1 achieved the smallest mean RMSE for both case studies, followed by TSVR-MPSO2. Large RMSE values for some RBF and MLFN experiments may be due to overfitting or divergence. Minimum and maximum RMSE, and the standard deviation of RMSE are also presented for all experiments. The TSVR-MPSO1 provided only slightly lower mean RMSE compared to most PSO methods. Yet, the advantage of TSVR-MPSO1 is more significant for

some experiments, and consistency is critical in SSE. For instance, case 17 in Table 6.14 indicates that the RMSE for TSVR-MPSO1 is 10% or lower than the other PSO methods. Fig. 6.9 presents the RMSE difference between SVR-ASPSO and TSVR-MPSO1 for the IEEE 118-bus case. Since the results are almost identical in most cases, the mean RMSE does not emphasize the RMSE differences observed in some experiments. Yet, in three experiments MSVR-MPSO1 provides a lower RMSE than SVR-ASPSO by 0.1. As the system size increases, it is expected that the RMSE values may increase. Thus, the RMSE differences may also become more significant.

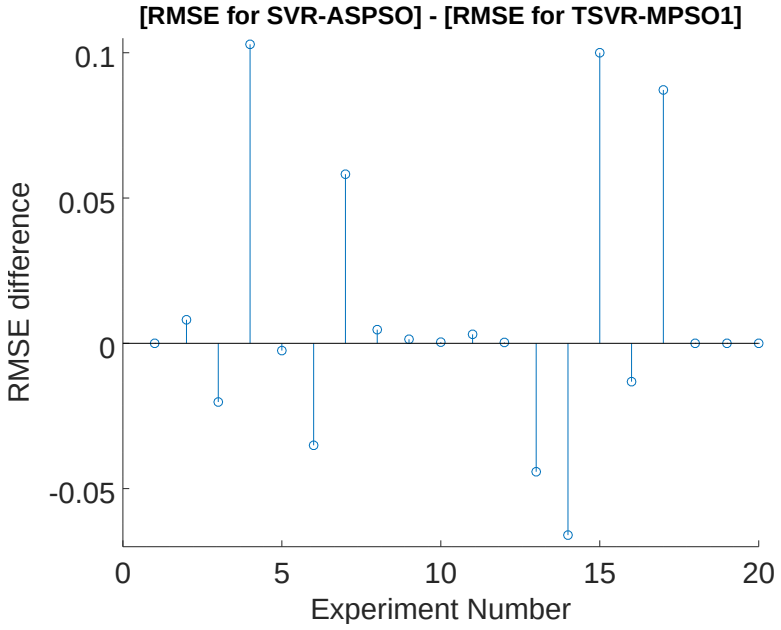


**Figure 6.9: RMSE comparison between SVR-PSO and TSVR-MPSO1. Positive values indicate higher RMSE for SVR-PSO.**

The optimized $\gamma, \epsilon$, and $C$ values of the best experiment for the IEEE 14-bus and 118-bus systems are presented in Tables 6.15 and 6.16, respectively. The best RMSE for the first four methods is identical for the IEEE 118-bus experiment, although the parameters in Table 6.16 are different. This is justified by the fact that these methods share the same SVR component, although their PSO component is different. For reference purposes, the SSI obtained by NRLF, TSVR-MPSO1, and RBF3 are shown in Table 6.14 for the IEEE 14-bus system.

**Table 6.13: Inertia weight and parameters of different methods**

| Method | Weight |
|---|---|
| **SVR-PSO** | $w_i^{t+1} = 0.95 w_i^t$ |
| **TSVR-MPSO1** | $w_i^{t+1} = 0.95 e^{-\lvert P_{i,Best}^t - G_{Best}^t \rvert^2}$ |
| **TSVR-MPSO2** | $w_i^{t+1} = 0.95 e^{-\lVert P_{i,Best}^t - G_{Best}^t \rVert^2}$ |
| **SVR-ASPSO** | $w_i^{t+1} = 0.95 e^{-\lVert P_{i,New}^t - X_{AS} \rVert^2}$ |
| PSO parameters for 14 Bus | $c_1 = c_2$=1.4962, Population: 20, Iterations: 20 |
| PSO parameters for 118 Bus | $c_1 = c_2$=1.4962, Population: 50, Iterations: 20 |
| **Case** | **Number of Neurons, Spread** |
| **RBF1** | 5, 0.1 |
| **RBF2** | 50, 0.1 |
| **RBF3** | 100, 0.1 |
| **Case** | **Number of Hidden Layers** |
| **MLFN1** | 5 |
| **MLFN2** | 10 |
| **MLFN3** | 20 |

**Table 6.14: One-line outage for IEEE 14-bus under 110% load base**

| Contingency | Line | SSI by NRLF | TSVR-MPSO | RBF |
|---|---|---|---|---|
| 1 | L 1-2 | 12.4809 | 12.4616 | 11.0306 |
| 2 | L 1-5 | 5.0828 | 4.9688 | 4.7325 |
| 3 | L 2-3 | 1.0004 | 1.1759 | 1.1308 |
| 4 | L 2-4 | 9.1830 | 7.9418 | 6.4777 |
| 5 | L 2-5 | 3.6291 | 3.7009 | 3.4460 |
| 6 | L 3-4 | 3.3366 | 3.4348 | 2.9732 |
| 7 | L 4-5 | 3.9656 | 3.8117 | 4.0995 |
| 8 | L 4-7 | 1.7433 | 1.6093 | 1.6213 |
| 9 | L 4-9 | 3.2589 | 3.4430 | 2.8741 |
| 10 | L 5-6 | 6.3977 | 6.3488 | 5.7640 |
| 11 | L 6-11 | 1.7091 | 1.7862 | 1.5930 |
| 12 | L 6-12 | 1.6126 | 1.5465 | 1.3774 |
| 13 | L 6-13 | 1.9205 | 2.0153 | 1.9517 |
| 14 | L 7-8 | 1.1943 | 1.1537 | 1.1985 |
| 15 | L 7-9 | 2.5523 | 2.7056 | 2.5459 |
| 16 | L 9-10 | 1.6029 | 1.5209 | 1.5777 |
| 17 | L 9-14 | 2.9751 | 2.9567 | 2.7368 |
| 18 | L 10-11 | 1.6200 | 1.6154 | 1.6507 |
| 19 | L 12-13 | 1.6167 | 1.6410 | 1.4888 |
| 20 | L 13-14 | 1.6459 | 1.6203 | 1.7579 |

**Table 6.15: Optimized SVR parameters of best experiment for IEEE 14-bus**

| Case | $\gamma$ | $\varepsilon$ | $C$ |
|---|---|---|---|
| **SVR-PSO** | 0.1174 | 1.5022e-4 | 5.6178e+03 |
| **TSVR-MPSO1** | 0.1124 | 1.4876e-04 | 5.6105e+03 |
| **TSVR-MPSO2** | 0.1167 | 1.4856e-04 | 6.3248e+03 |
| **SVR-ASPSO** | 0.1022 | 1.611e-04 | 5.5553e+03 |
| **SVR-GS** | 0.1459 | 1.2556e-04 | 5.6548e+03 |
| $X_{AS}$ | 0.2210 | 2.5786e-04 | 2.9747e+03 |

**Table 6.16: Optimized SVR parameters of best experiment for IEEE 118-bus**

| Case | $\gamma$ | $\varepsilon$ | $C$ |
|---|---|---|---|
| **SVR-PSO** | 0.1375 | 1.5058e-4 | 5.9193e+03 |
| **TSVR-MPSO1** | 0.1330 | 1.4914e-04 | 5.9403e+03 |
| **TSVR-MPSO2** | 0.1429 | 1.5574e-04 | 5.9720e+03 |
| **SVR-ASPSO** | 0.1425 | 1.4855e-04 | 5.9941e+03 |
| **SVR-GS** | 0.1695 | 1.3875e-04 | 5.0384e+03 |
| $X_{AS}$ | 0.2000 | 0.3661e-04 | 5.0168e+03 |

### 6.2.3 Simulation Results of Feature Selection by Multi-Objective PSO for SSE

In this section, the simulation results of different IEEE case studies are presented to show the effectiveness of our proposed method from section 5.3. The total number of generated data for each class, features, selected features, and the dimensionality reduction for each system are shown in Table 6.21. Tables. 6.22 and 6.23 shows the usefulness of feature selection in classification performances. SVM with polynomial kernel and SMO solver is used for all case studies in this table. The first row of Tables. 6.22 and 6.23 present the classification accuracy while all features are used. The second row shows the results of the SFS and the last row illustrates the classification accuracy by MOPSO. Both feature selection techniques improved the accuracy of the systems except for the IEEE 9-bus system which is a very small case study. However, MOPSO significantly improved the classification accuracy of other larger systems comparing SFS.

**Table 6.17: Prediction accuracy of different SVR methods for IEEE 14-bus system**

| RMSE | | | | | |
|---|---|---|---|---|---|
| **Exp** | **SVR-PSO** | **SVR-MPSO1** | **SVR-MPSO2** | **SVR-ASPSO** | **SVR-GS** |
| 1 | 0.0425 | 0.0398 | 0.0411 | 0.0409 | 0.0686 |
| 2 | 0.0634 | 0.0634 | 0.0744 | 0.0634 | 0.1111 |
| 3 | 0.0606 | 0.0606 | 0.0604 | 0.0606 | 0.0861 |
| 4 | 0.0394 | 0.0351 | 0.0381 | 0.0349 | 0.0921 |
| 5 | 0.0762 | 0.0762 | 0.0762 | 0.0771 | 0.1164 |
| 6 | 0.0233 | 0.0198 | 0.0233 | 0.0233 | 0.0661 |
| 7 | 0.0687 | 0.0694 | 0.0693 | 0.0704 | 0.0907 |
| 8 | 0.0667 | 0.0467 | 0.0494 | 0.0644 | 0.0981 |
| 9 | 0.0725 | 0.0725 | 0.0716 | 0.0725 | 0.0981 |
| 10 | 0.0291 | 0.0333 | 0.02601 | 0.0318 | 0.0759 |
| 11 | 0.0319 | 0.0319 | 0.0319 | 0.0319 | 0.0833 |
| 12 | 0.0438 | 0.0438 | 0.0438 | 0.0438 | 0.0766 |
| 13 | 0.0412 | 0.0413 | 0.0412 | 0.0419 | 0.0765 |
| 14 | 0.0474 | 0.0456 | 0.0441 | 0.0491 | 0.0966 |
| 15 | 0.0788 | 0.0807 | 0.0788 | 0.0799 | 0.1003 |
| 16 | 0.1048 | 0.1051 | 0.1036 | 0.1055 | 0.1373 |
| 17 | 0.0691 | 0.0676 | 0.0673 | 0.0677 | 0.1038 |
| 18 | 0.0803 | 0.0858 | 0.0815 | 0.0785 | 0.1284 |
| 19 | 0.0295 | 0.0234 | 0.0283 | 0.0269 | 0.0759 |
| 20 | 0.1136 | 0.1122 | 0.1136 | 0.1119 | 0.1254 |
| **Mean** | 0.0591 | **0.0577** | 0.0581 | 0.0591 | 0.0953 |
| **Min** | 0.0233 | 0.0198 | 0.0233 | 0.0233 | 0.0661 |
| **Max** | 0.1136 | 0.1122 | 0.1136 | 0.1119 | 0.1373 |
| **STD** | 0.0249 | 0.0259 | 0.0255 | 0.0249 | **0.0203** |

### 6.2.4 Simulation Results of Multi-Classifier Voter Model for Online SSE

In this section, the simulation results of different IEEE case studies are presented to show the effectiveness of our proposed method from section 5.4. Tables. 6.24-6.29 show the performance evaluation of different classification methods for IEEE 9-bus, 14-bus, 39-bus, 57-bus, 118-bus, and 300-bus. SVM-RBF-MPSO classifier gave the best correct accuracy for all case studies, and the voting technique improved the correct accuracies from 0.521% to 0.9976%.

**Table 6.18: Prediction accuracy of different ANN methods for IEEE 14-bus System**

| RMSE | | | | | | |
|---|---|---|---|---|---|---|
| **Exp** | **RBF1** | **RBF2** | **RBF3** | **MLFN1** | **MLFN2** | **MLFN3** |
| 1 | 1.2222 | 0.0923 | 0.0162 | 0.1502 | 0.1273 | 0.1181 |
| 2 | 1.2711 | 0.3839 | 0.1934 | 0.2221 | 0.1059 | 0.0914 |
| 3 | 1.3316 | 0.1629 | 0.0697 | 0.1066 | 0.1468 | 0.1309 |
| 4 | 1.3075 | 0.1497 | 0.0656 | 0.1231 | 0.1027 | 0.0694 |
| 5 | 1.3247 | 0.4892 | 0.2649 | 0.1059 | 0.1301 | 0.4543 |
| 6 | 1.2027 | 0.0828 | 0.0308 | 0.0317 | 0.1999 | 0.0591 |
| 7 | 1.3112 | 0.2396 | 0.0596 | 0.0557 | 0.0742 | 0.2501 |
| 8 | 1.2701 | 0.1636 | 0.0653 | 0.1675 | 0.0875 | 0.0701 |
| 9 | 1.3464 | 0.1841 | 0.0767 | 0.1761 | 0.1102 | 0.0996 |
| 10 | 1.2038 | 0.1052 | 0.0208 | 0.1159 | 0.1416 | 0.1428 |
| 11 | 1.2913 | 0.1409 | 0.0477 | 0.1297 | 0.0874 | 0.2998 |
| 12 | 1.1924 | 0.1463 | 0.0283 | 0.0646 | 0.0617 | 0.0651 |
| 13 | 1.3141 | 0.1165 | 0.0359 | 0.0204 | 0.2011 | 0.0695 |
| 14 | 1.2871 | 0.1222 | 0.0337 | 0.2084 | 0.3391 | 0.1292 |
| 15 | 1.3421 | 0.1412 | 0.0862 | 0.1494 | 0.1493 | 0.2961 |
| 16 | 1.4179 | 0.5258 | 0.3141 | 0.1211 | 0.1132 | 0.0841 |
| 17 | 1.1729 | 0.2209 | 0.0551 | 0.0871 | 0.1955 | 0.1636 |
| 18 | 1.3211 | 0.4488 | 0.3831 | 0.1752 | 0.2591 | 0.2833 |
| 19 | 1.3388 | 0.0863 | 0.0342 | 0.1488 | 0.1464 | 0.3114 |
| 20 | 1.4315 | 0.5076 | 0.2991 | 0.1314 | 0.2738 | 0.4586 |
| **Mean** | 1.2951 | 0.2255 | 0.1090 | 0.1245 | 0.1526 | 0.1823 |
| **Min** | 1.1729 | 0.0828 | **0.0162** | 0.0204 | 0.0617 | 0.0591 |
| **Max** | **1.4315** | 0.5258 | 0.3831 | 0.2221 | 0.3391 | 0.4586 |
| **STD** | 0.0699 | 0.1529 | 0.1138 | 0.05405 | 0.0723 | 0.1281 |

### 6.2.5 Simulation Results of GA Variant based Effective Solutions for ED

Three popular benchmarks, namely the IEEE 6-unit, 15-unit, and 10-unit systems, were used to evaluate the performance of GA variants. All algorithms were implemented in MATLAB for a population size of 50. The elite rate, crossover rate, and mutation rate were set to 10%, 80%, and 10%, respectively. Each algorithm was iterated for a maximum of $Eval_{max} = n \times 10^5$ function evaluations and was repeated 25 times.

For the 6-unit and 15-unit systems, the total load demand is 1263 MW and 2630 MW, respectively. For these two cases, power balance, generator limits, ramp rate limits, and POZ (*see* section 2.2) are used as constraints. The system coefficients, the line loss

**Table 6.19: Prediction accuracy of different SVR methods for IEEE 118-bus system**

| | RMSE | | | | |
|---|---|---|---|---|---|
| **Exp** | **SVR-PSO** | **TSVR-MPSO1** | **TSVR-MPSO2** | **SVR-ASPSO** | **SVR-GS** |
| 1 | 1.2336 | 1.2336 | 1.2275 | 1.2336 | 2.0910 |
| 2 | 0.9091 | 0.9010 | 0.9091 | 0.9091 | 1.0643 |
| 3 | 1.6491 | 1.6180 | 1.6221 | 1.5978 | 2.0104 |
| 4 | 0.8899 | 0.8871 | 0.8899 | 0.9899 | 1.7202 |
| 5 | 0.7867 | 0.7861 | 0.7831 | 0.7836 | 0.6561 |
| 6 | 0.4109 | 0.4161 | 0.3783 | 0.3810 | 0.4253 |
| 7 | 2.7824 | 2.7824 | 2.7723 | 2.8406 | 3.3024 |
| 8 | 1.7131 | 1.7015 | 1.6982 | 1.7062 | 1.8103 |
| 9 | 1.1137 | 1.1104 | 1.1103 | 1.1118 | 1.2522 |
| 10 | 0.9110 | 0.9106 | 0.9110 | 0.9110 | 0.9664 |
| 11 | 0.4911 | 0.4862 | 0.4865 | 0.4893 | 0.3188 |
| 12 | 1.8527 | 1.8523 | 1.8527 | 1.8526 | 1.0876 |
| 13 | 0.9981 | 0.8981 | 0.8581 | 0.8539 | 0.9626 |
| 14 | 0.6801 | 0.6800 | 0.5946 | 0.6140 | 0.7742 |
| 15 | 0.8617 | 0.8617 | 0.9617 | 0.9617 | 0.9638 |
| 16 | 0.7476 | 0.7546 | 0.7420 | 0.7414 | 0.7224 |
| 17 | 0.9979 | 0.8986 | 0.9803 | 0.9858 | 1.1529 |
| 18 | 0.7175 | 0.7175 | 0.7175 | 0.7175 | 0.7509 |
| 19 | 0.3318 | 0.3318 | 0.3418 | 0.3318 | 1.8419 |
| 20 | 0.2726 | 0.2726 | 0.2726 | 0.2726 | 0.6814 |
| **Mean** | 1.01753 | **1.0050** | 1.0054 | 1.01878 | 1.2277 |
| **Min** | 0.2726 | **0.2726** | 0.2726 | 0.2726 | 0.3188 |
| **Max** | 2.7824 | 2.7824 | 2.7723 | 2.8406 | 3.3024 |
| **STD** | 0.5989 | **0.5974** | 0.5994 | 0.5950 | 0.7132 |

coefficients, and the POZs are available in. For the 10-unit system, VPE and MFO are used as constraints. For this scenario, the load demand is 2700 MW. The system coefficients are available in section 6.1.2. Tables 6.30, 6.31, and 6.32 present the best, mean, median, and standard deviation of the cost provided by the GA variants for the three case studies.

Tables 6.33 and 6.34 presents detailed results regarding the best cost and the associated solution (unit powers) determined by the GA variants, and also by MSSA [50], ASPSO [106] and BSA [42] for the 6-unit case study. Similarly, Table 6.35 presents results for the GA variants, and Table 6.36 shows CBPSO-RVM [107], BSA [42] for the 15-unit case study. The convergence plots of the GA variants for this case study are illustrated in Figure 6.10. Finally, Tables 6.37 and 6.38 present results for the GA variants, as well

**Table 6.20: Prediction accuracy of different ANN methods for IEEE 118-bus system**

| RMSE | | | | | | |
|---|---|---|---|---|---|---|
| **Exp** | **RBF1** | **RBF2** | **RBF3** | **MLFN1** | **MLFN2** | **MLFN3** |
| 1 | 3.7056 | 1.9813 | 1.8358 | 1.3920 | 1.0295 | 1.4107 |
| 2 | 5.7756 | 20.220 | 21.2201 | 0.2844 | 25.399 | 17.8318 |
| 3 | 2.0890 | 9.1233 | 31.1233 | 16.549 | 8.9802 | 15.5829 |
| 4 | 1.0076 | 17.715 | 37.7153 | 0.1234 | 14.989 | 10.5833 |
| 5 | 1.7674 | 3.0245 | 11.0245 | 8.7696 | 8.4286 | 12.1043 |
| 6 | 1.7232 | 2.6608 | 34.6608 | 6.0969 | 14.104 | 14.9583 |
| 7 | 2.6644 | 5.4622 | 30.4622 | 15.593 | 2.5626 | 23.788 |
| 8 | 2.2309 | 5.9256 | 15.9256 | 6.8450 | 8.3807 | 17.0944 |
| 9 | 2.0457 | 6.8629 | 6.86290 | 10.309 | 10.761 | 10.7604 |
| 10 | 1.9480 | 5.7538 | 10.7538 | 7.5996 | 13.321 | 11.2232 |
| 11 | 1.1117 | 30.430 | 410.430 | 12.470 | 6.4015 | 38.739 |
| 12 | 2.8424 | 22.044 | 223.044 | 1.4053 | 9.2550 | 10.757 |
| 13 | 1.9291 | 16.697 | 168.697 | 15.861 | 21.519 | 18.0236 |
| 14 | 1.7823 | 8.8998 | 18.8998 | 9.3988 | 11.516 | 10.2112 |
| 15 | 1.8755 | 22.341 | 22.3414 | 10.927 | 11.332 | 8.6678 |
| 16 | 1.7004 | 19.546 | 19.5462 | 6.9739 | 13.473 | 25.3493 |
| 17 | 2.9949 | 29.403 | 292.403 | 8.9699 | 8.7041 | 17.4105 |
| 18 | 1.1939 | 14.053 | 14.0536 | 11.044 | 7.6404 | 15.9732 |
| 19 | 2.3743 | 26.738 | 26.7384 | 8.7102 | 8.2870 | 10.9883 |
| 20 | 3.9309 | 38.377 | 38.3775 | 4.5572 | 10.927 | 18.9669 |
| **Mean** | 2.3346 | 15.3631 | 71.8058 | 9.6940 | 11.8507 | 15.5212 |
| **Min** | 1.0076 | 1.9813 | 1.8358 | 1.3920 | 1.0295 | 1.4107 |
| **Max** | 5.7756 | 38.377 | **410.43** | 16.549 | 25.399 | 38.739 |
| **STD** | 1.1171 | 10.641 | 111.962 | 3.7368 | 5.2142 | 7.7076 |

**Table 6.21: Number of generated data and selected features**

| IEEE Test System | | | | | | |
|---|---|---|---|---|---|---|
| | 9-bus | 14-bus | 39-bus | 57-bus | 118-bus | 300-bus |
| Operating Scenario | 300 | 600 | 1000 | 1500 | 2000 | 3000 |
| Class Safe | 77 | 160 | 271 | 370 | 501 | 750 |
| Class Alarm | 81 | 155 | 251 | 366 | 556 | 802 |
| Class Insecure | 78 | 167 | 243 | 364 | 479 | 812 |
| Class Emergency | 64 | 118 | 235 | 400 | 464 | 636 |
| No. of Features | 17 | 36 | 91 | 159 | 283 | 734 |
| No. Selected Features | 4 | 5 | 6 | 8 | 10 | 12 |
| Dimensionality Reduction % | 23.53 | 13.89 | 6.59 | 5.03 | 3.53 | 1.63 |

**Table 6.22: Comparing MOPSO and SFS for IEEE 9-bus, 14-bus, and 39-bus**

| Classification Method: SVM-Polynomial CA% | | | |
|---|---|---|---|
| Feature Selection Method | IEEE 9-bus | IEEE 14-bus | IEEE 39-bus |
| No Feature Selection | **85.4387** | 81.8894 | 83.3345 |
| Sequential Forward Selection (SFS) | 82.2277 | 87.1254 | 89.4768 |
| Multi-Objective PSO (MOPSO) | 84.8423 | **90.4571** | **90.3267** |

**Table 6.23: Comparing MOPSO and SFS for IEEE 57-bus, 118-bus, and 300-bus**

| Classification Method: SVM-Polynomial CA% | | | |
|---|---|---|---|
| Feature Selection Method | IEEE 57-bus | IEEE 118-bus | IEEE 300-bus |
| No Feature Selection | 84.5763 | 82.7812 | 82.4355 |
| Sequential Forward Selection (SFS) | 85.4576 | 86.4212 | 84.3828 |
| Multi-Objective PSO (MOPSO) | **89.9872** | **89.2412** | **90.0985** |

as for CBPSO-RVM [107] and BSA [42], for the 10-unit system. The convergence plots of the GA variants for this case study are presented in Figure 6.11. The 10-unit case study took larger number of iterations to converge because it has multi-fuel options.

In summary, experimental results have revealed that KGA is the most consistent GA variant, both in terms of best cost and mean cost, although TRGA, FNGA, and UGA are also competitive. With respect to convergence, UGA was confirmed to be the fastest GA variant. In terms of the best cost, KGA outperformed BSA and CBPSO-RVM for the 10-unit case, and performed similarly to BSA and MSSA for the 6-unit case. BSA appears to exhibit a slight advantage over the GA variants for the 15-unit system.

**Table 6.24: Performance of multi-classifier voter for IEEE 9-bus**

| IEEE 9-bus | | | |
|---|---|---|---|
| Classifier Type | Training Time (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 0.4732 | 93.2245 | 85.4387 |
| SVM-Poly-ISDA | 1.9375 | 92.8212 | 85.2234 |
| SVM-Poly-L1QP | 0.3189 | 93.5232 | 85.4367 |
| SVM-RBF-MPSO1 | 39.7474 | 97.8344 | 86.6451 |
| SVM-RBF-MPSO2 | 48.4096 | 97.2352 | 86.6667 |
| SVM-RBF-MPSO3 | 49.4874 | 97.6756 | **86.9876** |
| RF-NO.TREE 30 | 0.34988 | 100.00 | 86.3461 |
| RF-NO.TREE 20 | 0.6192 | 99.5568 | 85.8972 |
| RF-NO.TREE 10 | 0.0941 | 96.1248 | 84.1241 |
| **Voting** | - | - | **87.9852** |
| **Improvement** | - | - | **0.9976** |

**Table 6.25: Performance of multi-classifier voter for IEEE 14-bus**

| IEEE 14-bus | | | |
|---|---|---|---|
| Classifier Type | Training Tim (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 0.1319 | 98.2543 | 90.3267 |
| SVM-Poly-ISDA | 0.1598 | 97.7556 | 90.3155 |
| SVM-Poly-L1QP | 0.2793 | 98.2543 | 90.3267 |
| SVM-RBF-MPSO1 | 13.034 | 96.5087 | **91.4577** |
| SVM-RBF-MPSO2 | 18.4414 | 95.5112 | 91.4572 |
| SVM-RBF-MPSO3 | 10.3262 | 96.5087 | 90.9547 |
| RF-NO.TREE 30 | 0.3980 | 99.7506 | 88.9246 |
| RF-NO.TREE 20 | 0.3336 | 100.00 | 87.9396 |
| RF-NO.TREE 10 | 0.1394 | 97.0074 | 81.4071 |
| **Voting** | - | - | **92.0255** |
| **Improvement** | - | - | **0.5678** |

**Table 6.26: Performance of multi-classifier voter for IEEE 39-bus**

| IEEE 39-bus | | | |
|---|---|---|---|
| Classifier Type | Training Time (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 0.6371 | 85.9821 | 90.0985 |
| SVM-Poly-ISDA | 3.5030 | 86.2265 | 90.0451 |
| SVM-Poly-L1QP | 0.9220 | 85.9005 | 90.0976 |
| SVM-RBF-MPSO1 | 156.0437 | 88.0195 | 90.1238 |
| SVM-RBF-MPSO2 | 166.8111 | 87.938 | 90.5612 |
| SVM-RBF-MPSO3 | 97.3404 | 87.8565 | **90.6712** |
| RF-NO.TREE 30 | 0.6316 | 100.00 | 83.8988 |
| RF-NO.TREE 20 | 0.8816 | 99.5110 | 81.2039 |
| RF-NO.TREE 10 | 0.2567 | 96.6585 | 82.2675 |
| **Voting** | - | - | **91.2156** |
| **Improvement** | - | - | **0.5444** |

**Table 6.27: Performance of multi-classifier voter for IEEE 57-bus**

| IEEE 57-bus | | | |
|---|---|---|---|
| Classifier Type | Training Time (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 0.6371 | 85.9821 | 90.0985 |
| SVM-Poly-ISDA | 3.5030 | 86.2265 | 90.0451 |
| SVM-Poly-L1QP | 0.9220 | 85.9005 | 90.0976 |
| SVM-RBF-MPSO1 | 156.0437 | 88.0195 | 90.1238 |
| SVM-RBF-MPSO2 | 166.8111 | 87.938 | 90.5612 |
| SVM-RBF-MPSO3 | 97.3404 | 87.8565 | **90.6712** |
| RF-NO.TREE 30 | 0.6316 | 100.00 | 83.8988 |
| RF-NO.TREE 20 | 0.8816 | 99.5110 | 81.2039 |
| RF-NO.TREE 10 | 0.2567 | 96.6585 | 82.2675 |
| **Voting** | - | - | **91.2156** |
| **Improvement** | - | - | **0.5444** |

**Table 6.28: Performance of multi-classifier voter for IEEE 118-bus**

| IEEE 118-bus | | | |
|---|---|---|---|
| Classifier Type | Training Time (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 1.2658 | 92.3771 | 89.2412 |
| SVM-Poly-ISDA | 1.9655 | 92.9381 | 89.2122 |
| SVM-Poly-L1QP | 1.5033 | 92.8838 | 89.2401 |
| SVM-PSO-TEST1 | 151.7228 | 96.8213 | **91.0461** |
| SVM-PSO-TEST2 | 202.1941 | 97.5313 | 90.1629 |
| SVM-PSO-TEST3 | 143.2213 | 96.8738 | 89.9678 |
| RF-NO.TREE 30 | 0.7719 | 100.00 | 90.2221 |
| RF-NO.TREE 20 | 1.3461 | 100.00 | 88.2168 |
| RF-NO.TREE 10 | 0.68296 | 98.2758 | 85.7713 |
| **Voting** | - | - | **91.5671** |
| **Improvement** | - | - | **0.5210** |

**Table 6.29: Performance of multi-classifier voter for IEEE 300-bus**

| IEEE 300-bus | | | |
|---|---|---|---|
| Classifier Type | Training Time (sec) | Train (CA%) | Test (CA%) |
| SVM-Poly-SMO | 1.2315 | 92.3538 | 89.9399 |
| SVM-Poly-ISDA | 1.6231 | 92.3538 | 90.0985 |
| SVM-Poly-L1QP | 1.41652 | 92.3538 | 89.9399 |
| SVM-PSO-TEST1 | 151.2341 | 96.5517 | 89.2882 |
| SVM-PSO-TEST2 | 202.2561 | 97.3763 | **91.1279** |
| SVM-PSO-TEST3 | 143.2415 | 96.6266 | 90.1381 |
| RF-NO.TREE 30 | 0.7452 | 100.00 | 89.9399 |
| RF-NO.TREE 20 | 1.0741 | 99.8501 | 87.3873 |
| RF-NO.TREE 10 | 0.2821 | 97.8261 | 85.2852 |
| **Voting** | - | - | **91.7165** |
| **Improvement** | - | - | **0.5886** |

**Table 6.30: Results of GA variants for 6-units**

| Method | Best ($/hr) | Mean ($/hr) | Med. ($/hr) | Std. |
|---|---|---|---|---|
| BGA | 15449.90979 | 15450.12343 | 15450.05756 | 0.193968201 |
| FNGA | 15449.96906 | 15451.09826 | 15450.77771 | 1.021224104 |
| KGA | **15449.89994** | **15449.92394** | **15449.92245** | **0.017044814** |
| TRGA | 15449.91319 | 15450.27628 | 15450.12628 | 0.32292185 |
| UGA | 15449.93556 | 15450.24261 | 15450.16914 | 0.25539524 |

**Table 6.31: Results of GA variants for 15-units**

| Method | Best ($/hr) | Mean ($/hr) | Med. ($/hr) | Std. |
|---|---|---|---|---|
| BGA | 32712.03 | 32715.95 | 32715.12 | 2.56917 |
| FNGA | 32706.7 | 32717.3 | 32714.79 | 8.086838 |
| KGA | **32704.81** | **32706.77** | 32706.49 | **1.593172** |
| TRGA | **32704.53** | 32707.32 | **32706.27** | 2.896268 |
| UGA | 32705.52 | 32708.3 | 32707.58 | 2.346371 |

**Table 6.32: Results of GA variants for 10-units**

| Method | Best ($/hr) | Mean ($/hr) | Med. ($/hr) | Std. |
|---|---|---|---|---|
| BGA | 623.9761 | 624.31287 | 625.13249 | 1.23677 |
| FNGA | 623.9432 | 624.11104 | 624.15378 | 0.1817 |
| KGA | **623.7736** | **623.85665** | **623.87061** | 0.09831 |
| TRGA | 623.8951 | 623.8879 | 623.90404 | 0.0855 |
| UGA | 623.8863 | 623.8819 | 623.88564 | **0.07753** |

**Table 6.33: Comparison results for the 6-unit system**

| Generation | BGA | FNGA | KGA | TRGA |
|---|---|---|---|---|
| P1 (MW) | 447.5560503 | 448.5251124 | 447.6611636 | 447.0228188 |
| P2 (MW) | 172.9237691 | 173.6749594 | 173.5994024 | 172.7145796 |
| P3 (MW) | 263.7451965 | 264.9752614 | 262.9094742 | 264.5056047 |
| P4 (MW) | 139.4709937 | 138.6599312 | 139.1772992 | 138.975938 |
| P5 (MW) | 164.8319461 | 164.8087921 | 165.5761515 | 165.7457395 |
| P6 (MW) | 87.41794403 | 85.32395972 | 87.03216628 | 87.00090973 |
| Total Generations (MW) | 1275.9459 | 1275.968016 | 1275.955657 | 1275.96559 |
| PL (MW) | 12.94541391 | 12.96764091 | 12.95558197 | 12.96539486 |
| Total generation cost ($/h) | 15449.90979 | 15449.96906 | **15449.89994** | 15449.91319 |

**Table 6.34: Comparison results for the 6-unit system**

| Generation | UGA | MSSA | APSO | BSA |
|---|---|---|---|---|
| P1 (MW) | 447.4926549 | 447.5029 | 446.66857 | 447.4902 |
| P2 (MW) | 173.8888666 | 173.3186 | 173.155594 | 173.3308 |
| P3 (MW) | 262.8151961 | 263.463 | 262.825958 | 263.4559 |
| P4 (MW) | 138.1555654 | 139.0656 | 143.468614 | 139.0602 |
| P5 (MW) | 166.1256904 | 165.473 | 163.91395 | 165.4804 |
| P6 (MW) | 87.50021122 | 87.1349 | 85.343745 | 87.1409 |
| Total Generations (MW) | 1275.978185 | 1275.958 | 1275.37643 | 1275.9583 |
| PL (MW) | 12.97662296 | 12.958 | 12.421628 | 12.9583 |
| Total generation cost ($/h) | 15449.93556 | **15449.8995** | 15449.99 | **15449.8995** |

**Table 6.35: Comparison results for the 15-unit system**

| Generation | BGA | FNGA | KGA | TRGA |
|---|---|---|---|---|
| P1 (MW) | 454.9993 | 455.0000 | 455.0000 | 455.0000 |
| P2 (MW) | 379.5807 | 380.0000 | 380.0000 | 380.0000 |
| P3 (MW) | 129.9073 | 129.9979 | 130.0000 | 130.0000 |
| P4 (MW) | 129.9933 | 129.9943 | 130.0000 | 130.0000 |
| P5 (MW) | 169.9111 | 168.8164 | 170.0000 | 169.9735 |
| P6 (MW) | 459.9703 | 459.9957 | 460.0000 | 460.0000 |
| P7 (MW) | 429.9738 | 430.0000 | 430.0000 | 430.0000 |
| P8 (MW) | 79.8044 | 61.96802 | 77.0887 | 69.3805 |
| P9 (MW) | 82.91083 | 70.05567 | 52.94777 | 61.29818 |
| P10 (MW) | 127.6979 | 160.0000 | 160.0000 | 159.9730 |
| P11 (MW) | 79.58085 | 79.99906 | 80.0000 | 80.0000 |
| P12 (MW) | 79.58233 | 79.75555 | 80.0000 | 80.0000 |
| P13 (MW) | 25.18185 | 25.0000 | 25.00248 | 25.02635 |
| P14 (MW) | 16.06463 | 15.0000 | 15.0000 | 15.0000 |
| P15 (MW) | 15.21293 | 15.11412 | 15.60567 | 15.0000 |
| Total Generations (MW) | 2660.371 | 2660.697 | 2660.645 | 2660.6520 |
| PL (MW) | 30.37138 | 30.69576 | 30.64399 | 30.65153 |
| Total generation cost ($/h) | 32712.03 | 32706.7 | **32704.81** | **32704.53** |

**Table 6.36: Comparison results for the 15-unit system**

| Generation | UGA | CBPSO-RVM | BSA |
|---|---|---|---|
| P1 (MW) | 454.9889 | 455.0000 | 455.0000 |
| P2 (MW) | 380.0000 | 380.0100 | 380.0000 |
| P3 (MW) | 130.0000 | 130.0000 | 130.0000 |
| P4 (MW) | 130.0000 | 126.5228 | 130.0000 |
| P5 (MW) | 169.9876 | 170.1312 | 170.0000 |
| P6 (MW) | 460.0000 | 460.0000 | 460.0000 |
| P7 (MW) | 430.0000 | 428.2836 | 430.0000 |
| P8 (MW) | 60.0000 | 60.0000 | 71.6368 |
| P9 (MW) | 77.11828 | 25.0000 | 59.0234 |
| P10 (MW) | 153.4569 | 159.7893 | 160.0000 |
| P11 (MW) | 80.0000 | 80.0000 | 80.0000 |
| P12 (MW) | 80.0000 | 80.0000 | 80.0000 |
| P13 (MW) | 25.0000 | 33.7037 | 25.0001 |
| P14 (MW) | 15.0000 | 55.0000 | 15.0001 |
| P15 (MW) | 15.0000 | 15.0000 | 15.0005 |
| Total Generations (MW) | 2660.552 | 2658.323 | 2660.661 |
| PL (MW) | 30.54953 | 28.36553 | 30.6609 |
| Total generation cost ($/h) | 32705.52 | 32976.68 | **32704.45** |

**Table 6.37: Comparison results for the 10-unit system**

| Generation | BGA | FNGA | KGA | TRGA |
|---|---|---|---|---|
| P1(MW) | 218.2629 | 218.1646 | 217.188 | 214.4453 |
| P2(MW) | 213.1529 | 209.9432 | 212.17 | 212.4232 |
| P3(MW) | 282.6736 | 283.5296 | 277.7917 | 278.8647 |
| P4(MW) | 241.2989 | 241.8433 | 239.4108 | 240.2289 |
| P5(MW) | 284.2452 | 281.9754 | 278.7414 | 276.1498 |
| P6(MW) | 236.8393 | 242.0798 | 240.4689 | 240.5948 |
| P7(MW) | 291.8698 | 287.0128 | 287.721 | 289.7905 |
| P8(MW) | 235.7892 | 242.7772 | 240.359 | 241.7059 |
| P9(MW) | 416.8246 | 423.4688 | 429.5014 | 427.3029 |
| P10(MW) | 279.0435 | 269.2094 | 276.6584 | 278.5045 |
| Total Generations(MW) | 2700.000 | 2700.004 | 2700.011 | 2700.010 |
| Total generation cost($/h) | 623.9761 | 623.9432 | **623.7736** | 623.8951 |

**Table 6.38: Comparison results for the 10-unit system**

| Generation | UGA | CBPSO-RVM | BSA |
|---|---|---|---|
| P1(MW) | 217.9038 | 219.2073 | 218.5777 |
| P2(MW) | 212.6456 | 210.2203 | 211.2153 |
| P3(MW) | 284.5811 | 278.5456 | 279.5619 |
| P4(MW) | 239.5394 | 239.3704 | 239.5024 |
| P5(MW) | 276.3653 | 276.412 | 279.9724 |
| P6(MW) | 237.1091 | 240.5797 | 241.1174 |
| P7(MW) | 290.8168 | 292.3267 | 289.7965 |
| P8(MW) | 239.5454 | 237.7557 | 240.5785 |
| P9(MW) | 427.7358 | 429.4008 | 426.8873 |
| P10(MW) | 273.7903 | 276.1815 | 272.7907 |
| Total Generations(MW) | 2700.033 | 2700.000 | 2700.0001 |
| Total generation cost($/h) | 623.8863 | 623.9588 | 623.9016 |

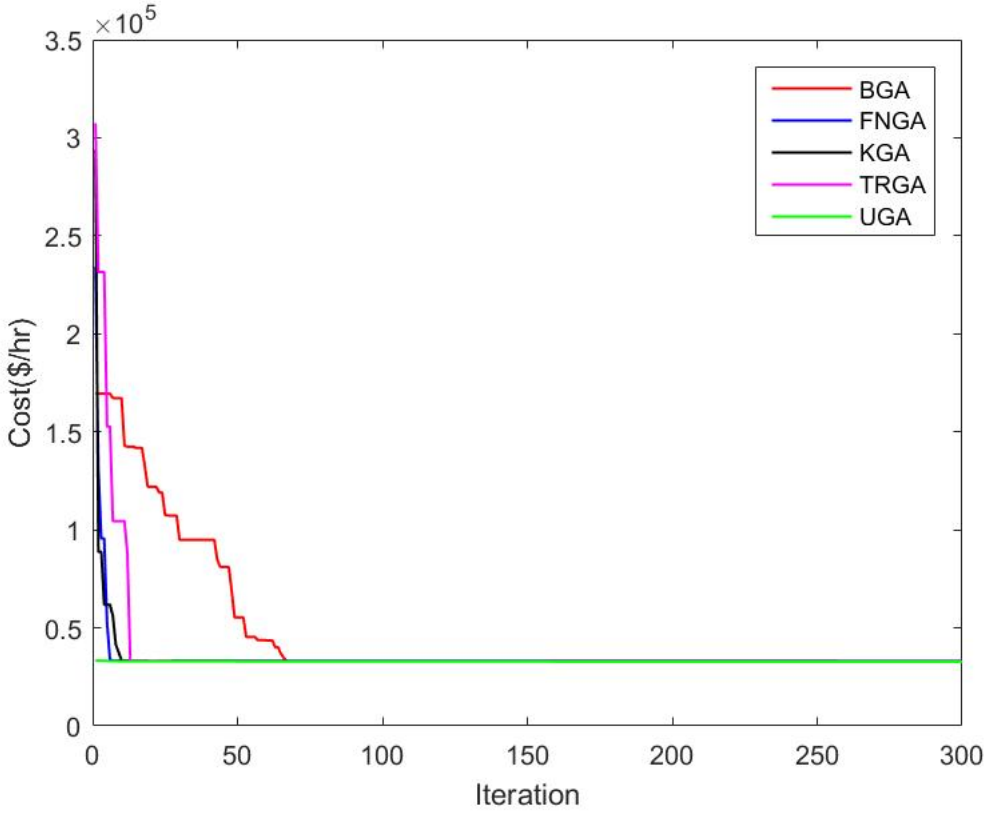


**Figure 6.10: Convergence plots of GA variants for 15-units**

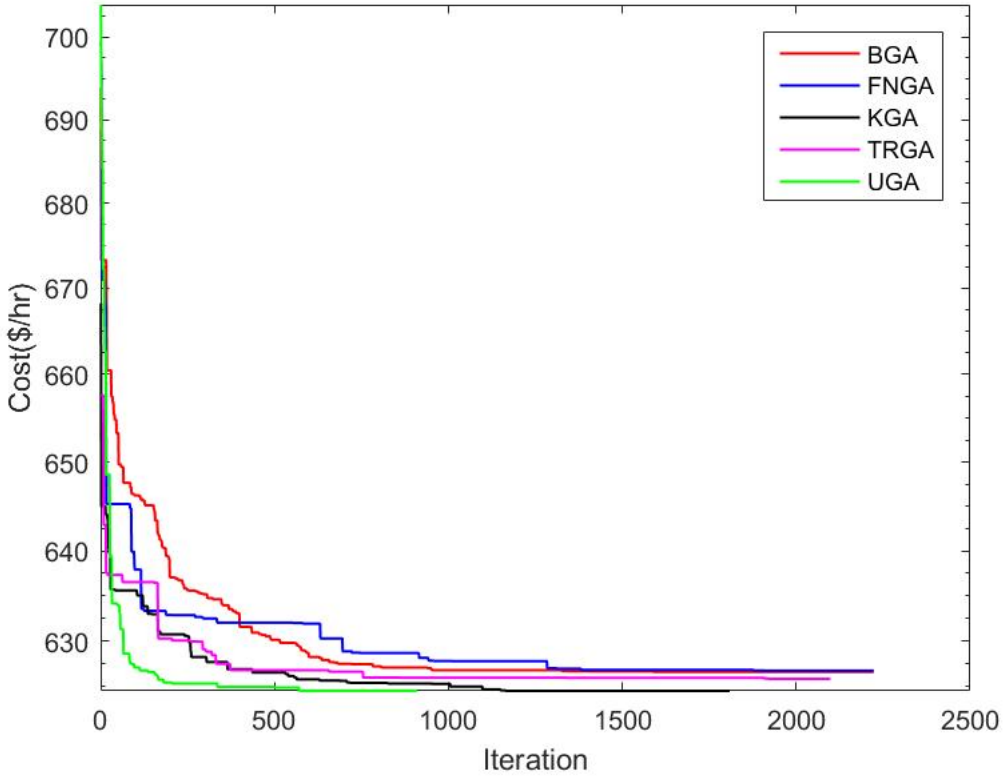


**Figure 6.11: Convergence plots of GA variants for 10-units**

# Chapter 7

# Conclusion and Future Work

This chapter presents some conclusions based on the proposed work. Moreover, some suggestions for future work regarding each of the proposed technique in chapter 5 are discussed.

## 7.1 Conclusions and Future Work for Section 5.1

Section 5.1, proposed a study of different optimization techniques which were used to obtain the optimal parameters of an SVM for static security assessment. As opposed to past works, this dissertation investigated the performance of optimization techniques considering a different number of classes, and thus, different security levels.

As expected, in general, a smaller number of classes results in higher CCR. Indeed, being able to differentiate between secure and insecure cases is an easier problem compared to being able to identify different levels of security. Nevertheless, it is not guaranteed that the same technique provides a consistently higher CCR in all cases. On the contrary, the preliminary results presented in this work may indicate otherwise. Yet, although a small difference in terms of CCR may still be proven crucial for the reliable operation of a system, the fact that all techniques exhibit a similar performance implies that additional studies are needed in order to verify this conclusion. On the other hand, in terms of execution times, HS followed by ACOr were consistently faster.

Future work includes investigation of additional optimization techniques, for a larger

number of trials, and for different bus systems. Also, different SSIs proposed in the literature could be considered. Such studies could be important for choosing the appropriate optimization technique depending on the security assessment conditions.

## 7.2 Conclusions and Future Work for Section 5.2

SSE is an essential problem in power systems which can be solved by machine learning methods. Their advantage over NRLF is that, once the training phase is complete, the SSI can be obtained almost instantaneously for different contingencies. Yet, the system topology should remain unaltered. Otherwise, the methods require retraining. SVR is a robust regression method which can be used to solve the SSE problem. However, the SVR parameters need to be tuned for good performance. In section 5.2, MPSO was used to set the SVR parameters. MPSO updates the inertial weight based on the exponential distance between the particle's best position and the global best position of the swarm. The experimental results demonstrate that the proposed TSVR-MPSO provides a slightly lower average RMSE with respect to SVR-ASPSO, without having to obtain an estimate of the best solution through analytical selection.

Future work includes improving our proposed MPSO to converge faster, and also to study the possibility of employing other evolutionary algorithms for tuning the SVR parameters. Also, considering larger case studies and using other proposed SSIs could be studied.

## 7.3 Conclusions and Future Work for Section 5.3

As mentioned in section 5.3, FS is an essential task which may be performed prior to classification to improve the correct accuracy and generalization of the classifier. Often, a large number of features may be responsible for reducing the correct classification performance of an algorithm, especially when some features associated with different classes do not possess any distinctive differences. In general, FS is a multi-objective problem. However, in most works presented in the power systems literature, it is treated

as a single-objective problem. In this work, the were two objectives considered. The first objective was minimizing the classification error rate, while the second objective was minimizing the number of features. We used MOPSO to select the features for solving the SSE problem. The experimental results have indicated that MOPSO performs better with respect to the classification rate than SFS for the same number of selected features. Moreover, the MOPSO technique exhibits a fast convergence rate. In future work, other multi-objective evolutionary algorithms (MOEAs) could be compared with MOPSO to solve the SSE problem.

## 7.4 Conclusions and Future Work for Section 5.4

In section 5.4, a multi-classifier voting model was proposed for online SSE. Several multi-classifier models were trained, and a simple voting technique was applied to the output of all models. The class label with the largest number of votes was selected as the output. The proposed SVM-RBF-MPSO classifier provided the best correct accuracy for all case studies, and the voting technique improved the correct accuracies from 0.521% to 0.9976% for the different case studies. Future work includes checking other voting techniques and more multi-classifier models.

## 7.5 Conclustion and Fucture Work of Section 5.5

In section 5.5, several GA variants were employed to solve the ED problem. All GA variants, as well as other algorithms tested, provided a seemingly similar best cost. Nevertheless, even small savings of $0.1-0.2/h$ for a 10-unit system may result in considerable savings for a significantly larger system over a long period of time. In general, KGA appears to be the most consistent of the GA variants for this problem, both in terms of the best cost as well as the mean cost. The GA variants tested in this work were proven to be strong competitors to other ED solutions such as MSSA and BSA. Future work includes testing other GA variants for our case studies, but also for larger case studies.

# Bibliography


[1] Edited by Kwang Y. Lee and Mohamed A. El-Sharkawi. Modern heuristic optimization techniques : theory and applications to power systems. Hoboken, N.J. :Wiley ; 2008. Print.

[2] S. Rastgoufard, D. Charalampidis, "Parameter selection of multi-class SVM with evolutionary optimization methods for static security evaluation in power systems," *IEEE Electrical Power and Energy Conference (EPEC)*, 2016.

[3] S. Rastgoufard, D. Charalampidis, "Tuned support vector regression by modified particle swarm optimization for online power system static security evaluation," *IEEE Texas Power and Energy Conference (TPEC)*, 2018.

[4] S. Rastgoufard, S. Iqbal, MD. T. Hoque, D. Charalampidis, "Genetic algorithm variant based effective solutions for economic dispatch problems", *IEEE Texas Power and Energy Conference (TPEC)*, 2018.

[5] S. Kalyani, K.S. Swarup, "Particle swarm optimization based K-means clustering approach for security assessment in power systems,", *Expert Systems with Applications*, vol. 38, pp. 10839-10846, 2011.

[6] K. Niazi, C. Arora, S. Surana, "Power system security evaluation using ANN: Feature selection using divergence," *Electric Power Systems Research*, vol. 69, pp. 161-167, 2004.

[7] K. Morison, L. Wag, P. Kundur, "Power system security assessment," *IEEE Power and Energy Mag,* vol. 2, no. 5, pp. 30-39, Oct. 2005.

[8] N. Tomin, V. Kurbatsky, D. Sidorov, " Machine learning techniques for power system security assessment," *IFAC*, vol. 49-27, pp. 445-450, 2016.

[9] P. Sekhar, S. Mohanty, "An online power system static security assessment module using multi-layer perceptron and radial basis function network,", *Electrical power and energy systems*, vol. 76, pp. 165-173, 2016.

[10] SM. Shahidipour, "Communication and control in electric systems", Wiley interscience, John Wiley and Sons, 2003.

[11] Y. Mansour, E. Vaahedi, M. El-Sharkawi, " Neural network based pattern recognition for power system security assessment," *IEEE Trans. Neural Network,* vol. 8, no. 4, pp. 942-950, 1997.

[12] S. Kalyani, K.S. Swarup, "Pattern analysis and classification for security evaluation in power networks." *Electrical Power and Energy Systems*, vol. 44, pp. 547-560, 2013.

[13] Sunitha. R, R.S. Kumar, A.T. Mathew, "Online static security assessment module using artificial neural networks.", *IEEE Trans. on Power Systems*, vol. 28, no.4, 2013.

[14] R. Schainker, P. Miller, W. Dubbelday, P. Hirsch, and G. Zhang, "Real-time dynamic security assessment: Fast simulation and modeling applied to emergency outage security of electric grid," *IEEE Power and Energy Mag.*, vol. 4, no. 2, pp. 51-58, 2006.

[15] D.S. Javan, H. R. Mashhadi, M. Rouhani, "A fast static security assessment method based on radial basis function neural networks using enhanced clustering," *Electrical power and energy systems*, vol. 44, pp. 988-996, 2013.

[16] S. Kalyani, K.S. Swarup, "Classifier design for static security assessment using particle swarm optimization", *Applied Soft Computing,* vol.11, pp. 658-666, 2011.

[17] S. Kalyani, K.S. Swarup, "Static security assessment in power systems using multiclass SVM with parameter selection methods", *International journal of computer theory and engineering,* vol. 5, no. 3, June 2013.

[18] I.S. Saeh, M.W. Mustafa, Y.S. Mohammed, M. Almaktar, "Static security classification and evaluation classifier design in electric power grid with presence of PV power plants using C-4.5", *Renewable and Sustainable Energy Reviews.* vol. 56, PP. 283-290, 2016.

[19] L. Hassan, M. Moghavvemi M, H. Almuriv, O. Steinmayer, "Current state of neural networks applications in power systems monitoring and control," *International Journal of Electr power and eneregy syst*, vol. 51, pp. 134-144, Oct. 2013.

[20] V. Vankayala, N. Rao, "Artificial neural network and their application to power system," *Electric Power Systems Research* vol. 28, no. 1, pp. 67-79, Oct. 1993.

[21] K. Swarup, P. Corthis, "ANN approach assesses system security," *IEEE Computer Applications in Power,* vol. 15, no. 3, PP. 421-426, Jul. 2002.

[22] T. Sidhu TS, C.Lan, "Contingency screening for steady-state security analysis by using FFT and artificial neural networks," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 1253-1263, Feb. 2000.

[23] R. Fischl, "Application of neural networks to power system security: technology and trends," *IEEE World Congress on Computational Intelligence*, Jul. 1994.

[24] W. Zhao, T. Tao, E. Zio, W. Wang, "A novel hybrid method of parameters tuning in support vector regression for reliability prediction: Particle swarm optimization combined with analytical selection," *IEEE Transactions on Reliability*, vol. 65, no. 3, pp. 1393-1405, Sept. 2016.

[25] J.C. Bansal, P.K. Singh, M. Sraswat, "Inertia weight strategies in particle swarm optimization", *IEEE Nature and biologically inpired computing,* 2011.

[26] C.C. Chang and C.J. Lin, "Training v-support vector classifiers: Theory and algorithms," *Neural Computat.*, vol. 13, no. 9, pp. 2119-2147, Sep. 2001.

[27] C.W. Hsu, C.C. Chang and C.J. Lin, *A practical guide to support vector classification*, 2003.

[28] S.W. Lin, K.C. Ying, S.C. Chen, and Z.J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Syst. Applic*, vol. 35, no. 4, pp 1817-1824, Nov. 2008.

[29] Z.L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Trans. Power Syst.* vol. 18, no. 3, pp. 1187-1195, 2003.

[30] D.S. Broomhead, D. Lowe "Radial basis functions, multi-variable functional interpolation and adaptive networks." *Technical report. RSRE,* 1988.

[31] P. Auer, H.Burgsteiner, W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons". *Neural Networks*, vol. 21, no. 5, pp. 786–795, 2008.

[32] L.A. Zadeh, "Fuzzy sets", *Information and Control,* vol. 8, no. 3,pp.338-353, 1965.

[33] C. Cortes and V. Vapnik, Machine learning, 1995.

[34] J.R. Quinlan, "Simplifying decision trees",*International Journal of Man-Machine Studies*, vol. 27, no. 3, 1987.

[35] J.A. Hartigan, M.A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm", emphJournal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pp. 100-108, 1979.

[36] L.P. Kaelbling, M.L. Littman, M.W. Andrew W, "Reinforcement Learning: A Survey",*Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[37] V. Jadoun, N. Gupta, K.R. Niazi, A. Swarnkar, "Modulated particle swarm optimization for economic emission dispatch," *Electr. Power Energy Syst.* vol. 73, pp. 80-88, 2015.

[38] J. Dodu, P. Martin, A. Merlin, J. Pouget, "An optimal formulation and solution of short-range operating problems for a power system with flow constraints," *Proc. IEEE*, vol. 60, no. 2, pp. 54-63, 1972.

[39] J. Y. Fan, L. Zhang, "Real-time economic dispatch with line flow and emission constraints using quadratic programming," *IEEE Trans. Power Syst.*, vol. 13, no. 2, pp. 320-325, 1998.

[40] R. M. Chen, "Application of the fast Newton-Raphson economic dispatch and reactive power/voltage dispatch by sensitivity factors to optimal power flow," *IEEE Trans. Energy Convers*, vol. 10, no. 2, pp. 293-301, 1995.

[41] D.C. Walters, G.B. Sheble, "Genetic algorithm solution of economic dispatch with valve point loadings," *IEEE Trans. Power Syst.*, vol. 8, no.4, pp. 1325-1332, 1993.

[42] M. Modiri-Delshad, S. Hr. Aghay Kaboli, E.Taslimi-Renani, N. Abd Rahim, "Backtracking search algorithm for solving economic dispatch problems with valve-point effects and multiple fuel options," *Energy*, vol. 116, pp. 637-649, 2016.

[43] C.L. Chiang, "Genetic-based algorithm for power economic load dispatch," *IET Gen. Transm. Distrib.*, vol. 1, no. 2, pp. 261-269, 2007.

[44] W.M. Lin, F.S. Cheng, M.T. Tsay, "An improved Tabu search for economic dispatch with multiple minima," *IEEE Trans. Power Syst.*, vol. 17, no. 1, pp. 108-112, 2002.

[45] N. Noman, H. Iba, "Differential evolution for economic load dispatch problems," *Electr. Power Syst. Res.*, vol. 78, no. 3, pp. 1322-1331, 2008.

[46] S. Pothiya, I. Ngamroo, W. Kongprawechnon, "Ant colony optimization for economic dispatch problem with non-smooth cost functions," *Int. J. Electr. Power Energy Syst.*, vol. 32, no. 5, pp. 478-87, 2010.

[47] M. Fesanghary, MM. Ardehali, "A novel meta-heuristic optimization methodology for solving various types of economic dispatch problem," *Energy*, vol. 34, no. 6, pp. 757-766, 2006.

[48] Y. Labbi, D. Attous, B. Mahdad, "Artificial bee colony optimization for economic dispatch with valve point effect," *Frontiers in Energy*, vol. 8, no. 4, pp. 449-458, 2014.

[49] J. Yu and V. Li, "A social spider algorithm for solving the non-convex economic load dispatch problem," *Neurocomputing*, vol. 171, pp. 955-965, 2016.

[50] W.T. Elsayed, Y.G. Hegazy, F.M. Bendary, M.S. El-bages, "Modified social spider algorithm for solving the economic dispatch problem," *Engineering Science and Technology, an International Journal,* vol. 19, no. 4, pp. 1672-1681, 2016.

[51] S. Duman, N. Yorukeren, IH. Altas, "A novel modified hybrid PSOGSA based on fuzzy logic for non-convex economic dispatch problem with valve-point effect," *International Journal of Electrical Power & Energy Systems*, Vol. 64, pp. 121-135, 2015.

[52] A. Bhattacharya, P.K. Chattopadhyay, "Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch," *IEEE Trans. Power Syst,* vol. 25, no. 4, pp. 1955-1964, 2010.

[53] B. Mahdad, K. Srairi, "Interactive gravitational search algorithm and pattern search algorithms for practical dynamic economic dispatch," *Int. Trans. Electr. Energy Syst.*, vol. 25, pp. 2289-2309, 2015.

[54] S. Iqbal, M.T. Hoque, "AMLGA: A Genetic Algorithm with Adaptive and Memory-Assisted Local Operators", *Technical paper TR2016/2*, `http://cs.uno.edu/~tamjid/TechReport/AMLGA_TR20162.pdf`.

[55] J. Holland, "Adaptation in natural and artificial systems: Introductory analysis with applications to biology, control, and artificial intelligence", 1992.

[56] H. M$'$uhlenbein and D. Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm (BGA)," *Evolutionary Computation*, vol. 1, no. 1, pp. 25-49, 1993.

[57] A. Ukil, Intelligent systems and signal processing in power engineering. Berlin: Springer Verlag, 2007.

[58] M.V. Suganyadevi, C.K. Babulal, S. Kalyani, "Assessment of voltage stability margin by comparing various support vector regression models," *Soft Computing.* vol. 20, no. 2, pp. 807-818, 2016.

[59] `http://www.ra.cs.uni-tuebingen.de/lehre/ss12/advanced_ml/lecture6.pdf`

[60] `http://www.ra.cs.uni-tuebingen.de/lehre/ss12/advanced_ml/lecture6.pdf`

[61] V. Vapnik, The nature of statistical learning theory. Springer, New York, 1995.

[62] V. Kecman, T. M. Huang, M. Vogt. "Iterative single data algorithm for training kernel machines from huge data sets: Theory and Performance." *In Support Vector Machines: Theory and Applications,* pp. 255-274. Berlin: Springer-Verlag, 2005.

[63] R.E. Fan, P.H. Chen, C.J. Lin. "Working set selection using second order information for training support vector machines." *Journal of Machine Learning Research*, vol. 6, pp. 1889-1918, 2005.

[64] S. Escalera, O. Pujol and P. Radeva, "On the decoding process in ternary error-correcting output codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 120-134, 2010.

[65] J. Tin-Yau Kwok, "Support vector mixture for classification and regression problems," *Pattern Recognition, Fourteenth International Conference*, Aug. 1998.

[66] D. Mattera and S. Haykin, "Support vector machines for dynamic reconstruction of a chaotic system," *Advances in Kernel Methods*, pp. 211-242, 1999.

[67] A. Smola, B. Scholkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199-222, Aug. 2004.

[68] Ho, Tin Kam (1998). "The random subspace method for constructing decision forests".*IEEE Transactions on Pattern Analysis and Machine Intelligence.* vol. 20, no. 8, pp. 832-844, 1998.

[69] L. Breiman, "Bagging predictors". *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.

[70] T. Hastie, R. Tibshirani, J. Friedman, "The Elements of Statistical Learning", Springer, 2009.

[71] L. Breiman, Random Forests. *Machine Learning.* vol. 45, pp. 5-32, 2001.

[72] G. James, D. Witten,T. Hastie, R. Tibshirani, "An Introduction to Statistical Learning", Springer. pp. 316-321, 2013.

[73] A.M. Cedeno, J.Q. Dominguez, M.G. Corthina. D. Andina, "Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network", *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 2010.

[74] D.S. Broomhead, D. Lowe, "Multivariable functional interpolation and adaptive networks". *Complex Systems.* vol. 2, pp. 321–355, 1988.

[75] F. Schwenker, H.A. Kestler, G. Palm, "Three learning phases for radial-basis-function networks.", *Neural Networks*, vol. 14, pp. 439-458, 2001

[76] A. Bianconi, C.J. VonZuben, A.B. Serapiao, and J.S. Govone. "Artificial neural networks: A novel approach to analyzing the nutritional ecology of a blowfly species", *Journal of Insect Science.* vol. 10, no. 58. 1536-2442, 2010.

[77] B. Xue, M. Zhang, W.N. Browne, X. Yao, "A survey on evolutionary computation approches to feature selection," *IEEE Transaction on Evolutionary Computation*, vol. 20, no. 4, August 2016.

[78] M. Dash, H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 1-4, pp. 131-156, 1997.

[79] B. Xue, M. Zhang, W.N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach,", *IEEE Trans. on Cybernetics*, vol.43, no.6, 2013.

[80] H. Back, D.B. Fogel, and Z. Michalewicz (Editors), :Handbook of Evolutionary Computation", *Oxford University Press*, 1997.

[81] G. Beni, J. Wang, "Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems," *Lecture Notes in Computer Science*, vol. 3342. 2005.

[82] R. Storn, K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[83] L. Arya, S.Choube, R.Arya. "Differential evolution applied for reliability optimization of radial distribution systems," *Int. J. Electr. Power Energy Syst.*, vol. 33, no. 2, pp. 271–277, 2011.

[84] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research,* vol. 185, no. 3, pp. 1155-1173, 2008.

[85] L. Pereira, J. Papa and A. de Souza, "Harmony search applied for support vector machines training optimization," *Eurocon IEEE*, 2013.

[86] X. Yang, "Music-Inspired Harmony Search Algorithm", *The series Studies in Computational Intelligence*, vol. 191, pp. 1-14, 2009.

[87] R.D. Zimmerman, C.E.Murillo-Sanchez, "MatPower 5.1 User's Manual", `http://www.pserc.cornell.edu/matpower/manual.pdf`

[88] H. M´uhlenbein and D. Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm (BGA)," *Evolutionary Computation*, vol. 1, no. 1, pp. 25-49, 1993.

[89] H. M'uhlenbein and D. Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm (BGA)," *Evolutionary Computation*, vol. 1, no. 4, pp. 335-60, 1993.

[90] M.T. Hoque and S. Iqbal, "Genetic Algorithm based Improved Sampling for Protein Structure Prediction," *International Journal of Bio-Inspired Computation*, vol. 9, no. 3, pp. 129-141, 2017.

[91] M.T. Hoque, M. Chetty, L. Dooley, "Critical Analysis of the Schemata Theorem: The Impact of Twins and the Effect in the Prediction of Protein Folding Using Lattice Model," *GSIT, MONASH University*, 2005.

[92] M.T. Hoque, M. Chetty, L. Dooley, "Generalized schemata theorem incorporating twin removal for protein structure prediction," *Pattern Recognition in Bioinformatics*, pp. 84-97, 2007.

[93] M.T. Hoque, M. Chetty, A. Lewis, A. Sattar, "Twin removal in genetic algorithms for protein structure prediction using low-resolution model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 1, pp. 234-245, 2011.

[94] S. Iqbal, and M.T. Hoque MT, "A Homologous Gene Replacement based Genetic Algorithm," *Proceedings of Genetic and Evolutionary Computation Conference Companion*, ACM, 2016.

[95] S. Iqbal S, and M.T. Hoque MT, "hGRGA: A scalable genetic algorithm using homologous gene schema replacement," *Swarm and evolutionary computation*, vol. 34, pp. 33-49, 2017.

[96] M.O. Hassan, S.J. Cheng and Z.A. Zakaria, "Steady-State Modeling of Static Synchronous Compensator and Thyristor Controlled Series Compensator for Power Flow Analysis." *Information Technology Journal*, Vol. 8, pp. 347-353, 2009.

[97] G. Pranava, P.V. Prasad, "Constriction coefficient particle swarm optimization for economic load dispatch with valve point loading effect," *Power, Energy and Control (ICPEC)*, 2013.

[98] C.A. Coello, G.T. Pulido, M. S. Lechuga, "Handling multiple objectives with particle swarm optimization,", *IEEE Transactions on Evolutionary Computation*, vol. 8, no.3, 2004.

[99] J. D. Knowles, D.W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, pp. 149–172, 2000.

[100] J. D. Knowles and D.W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.,* vol. 8, pp. 149-172, 2000.

[101] N. Azadani, E. Hosseinian, S. Hasanpor, "Stability Constrained Optimal Power Flow in Deregulated Power Systems", *Electric Power Components and Systems.* vol.39, pp. 713-732, 2011.

[102] D. Subhasish, G. Sadhan, G. Arup, "Congestion management considering wind energy sources using evolutionary algorithm. *Electric Power Components and Systems.*, vol. 43, no. 7, 2015.

[103] A.A. El-Sawy, Z.M. Hendawy, "Reference Point Base TR-PSO for Multi-Objective Environmental/Economic Dispatch", *Applied Mathematics,* vol. 4, no. 5, 2014.

[104] `http://motor.ece.iit.edu/data/IEEE118bus_inf/IEEE118bus_figure.pdf`

[105] `http://al-roomi.org/power-flow/300-bus-system`

[106] B.K. Panigrahi, V. Ravikumari, Sanjoy Das, "Adaptive particle swarm optimization approach for static and dynamic economic load dispatch," *Energy Conversion and Management*, vol. 49, pp. 1407-1415, 2008.

[107] H. Lu, P. Sriyanyong, YH. Song, "Experimental study of a new hybrid PSO with mutation for economic dispatch with non-smooth cost function," *Int. J. Electr. Power Energy Syst.*, vol. 32, no. 9, pp. 921-935, 2010.

# Vita

The author was born in December, $17^{th}$ 1986 in Shiraz, Iran. She obtained her Bachelor's and Master's degree in Electrical Engineering (Control) from Shiraz University, Iran in 2009 and 2012 respectively. She joined the University of New Orleans (UNO) in August 2013 as a Ph.D. candidate in the Electrical Engineering (EE) department. She worked as a teacher and research assistant in EE department of UNO from August 2013 till May 2017 under the supervision of Dr. Dimitrios Charalampidis. She passed her qualifying exam, general exam, and prospectus exam in April 2015, January 2016, and January 2018, respectively. During her Ph.D. career, she published three conference papers and submitted one journal paper to IEEE. She started to work at Entergy's Corp. as a Co-Op since March 2016 in transmission relay setting and relay design groups. Her research interests include artificial intelligence, machine learning, data mining, evolutionary computation, and optimization techniques.