A Comparative Study of Optimization Techniques for Minimizing Fuel Cost in the Smart Grid

دراسة مقارنة لتقنيات التحسين لتقليل تكلفة الوقود في الشبكات الذكية

A Thesis Submitted as partial fulfillment of requirements of the degree of
Doctor of Philosophy in Computer Science

by

Alaa Siddig Ali Mohammed Ahmed ALhuruk

Supervisor:

Prof. Dr. Alaa Sheta

September 2022

Dedication

The sake of Allah, my Creator and my great teacher and messenger, Mohammed (May Allah bless and grant him), who taught us the purpose of life.

Acknowledgements

Firstly, I thank God who helped me and, I would like very much to thank my supervisor, Prof. Alaa Sheta, for his invaluable supervision and warm-hearted encouragement throughout this thesis. Without him, this thesis would never have happened. He always supported my scientific endeavors and was, and will be, the source of inspiration for both my research and my life in particular. Finally, I would like to express my gratitude to my husband and my family for their continuing support and love, and knowing that they are always there for me in any situation, made this experience, and makes my life, a lot easier.

Abstract

The design of a smart electric power grid is a challenge. One of the famous problems in this field is the Economic load dispatch (ELD) problem. ELD is a challenge optimization problem to minimize the total cost of the thermally generated power that satisfies a set of equality and inequality constraints. To solve this problem, we need to maximize the power network load under several operational constraints. Meanwhile, we need to minimize the cost of power generation and minimizing the loss in the network transmission. Traditional optimization methods were used to solve such problems as linear programming. Meta-heuristic search algorithms have shown encouraging performance in solving various real-life complex problems. This thesis attempts to provide a comprehensive comparison between nine meta-heuristic search algorithms including Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Crow Search Algorithm (CSA), Differential Evolution (DE), Salp Swarm Algorithm (SSA), Harmony Search (HS), Sine Cosine Algorithm (SCA), Multi-Verse Optimizer (MVO), and Moth-Flame Optimization Algorithm (MFO). Our developed results demonstrated that meta-heuristics search algorithms (i.e., CSA and DE) can offer the optimal set of power for each power station. These are computed power fulfill the supply needs and maintain both minimum power cost and minimum power losses in power transmission. In the future, we hope to continue to solving the power generation problem area like unit commitment problems by apply on Meta-heuristics algorithm and explores the best minimums fuel cost.

المستخلص

يمثل تصميم شبكة الطاقة الكهربائية الذكية تحديًا. واحدة من المشاكل الشهيرة في هذا المجال هي مشكلة إرسال الحمل الاقتصادي (ELD). ELD هي مشكلة تحسين التحدي لتقليل التكلفة الإجمالية للطاقة المولدة حراريًا والتي تلبي مجموعة من قيود المساواة وعدم المساواة. لحل هذه المشكلة، نحن بحاجة إلى زيادة تحمل شبكة الطاقة في ظل العديد من القيود التشغيلية. في غضون ذلك، نحن بحاجة لتقليل تكلفة توليد الطاقة وتقليل الخسارة في نقل الشبكة. تم استخدام طرق التحسين التقليدية لحل مشاكل مثل البرمجة الخطية. أظهرت خوارزميات البحث الفوقي أداءً مشجعًا في حل العديد من المشكلات المعقدة الواقعية. تحاول هذه الاطروحة تقديم مقارنة شاملة بين تسعة خوارزميات للبحث الفوقية وهي الخوارزميات الجينية (GAs) ، تحسين حشد الجسيمات (PSO) ، خوارزمية البحث عن الغراب (CSA) ، التطور التفاضلي (DE) ، خوارزمية سرب سرب (SSA) ، التناغم البحث (HS) وخوارزمية جيب التمام الجيبي (SCA) ومحسن الآيات المتعددة (MVO) وخوارزمية تحسين لهب العثة (MFO). أظهرت نتائجنا المطورة أن خوارزميات البحث الفوقية (أي CSA و DE) يمكن أن تقدم مجموعة الطاقة المثلى لكل محطة طاقة. هذه طاقة محسوبة تفي باحتياجات الإمداد وتحافظ على الحد الأدنى من تكلفة الطاقة والحد الأدنى من خسائر الطاقة في نقل الطاقة. في المستقبل، نأمل أن نستمر في حل مشكلة توليد الطاقة مثل مشاكل التزام الوحدة من خلال التطبيق على خوارزمية الاستدلال الفوقي واستكشاف أفضل تكلفة لوقود.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preface

The main aim of power system supply utility has been identified to provide a smooth power generation system to the consumers. It will be ensured that the electrical power is generated with minimum cost. That is mean to achieve an economic operation of the power system; the total demand must be appropriately shared among the units. This will minimize the total generation cost for the power system with the voltage level maintained at the safe operating limits. Economic dispatcher defined as the process of allocating generation levels to the generating units in the mix so that the system load is fully supplied most economically. The method of economic dispatch for generating units at different loads must have total fuel cost at the minimum point.

Meta-heuristics are global search algorithms and their goal is to find an acceptable solution within a reasonable time frame when the problem is very complex and the search space is extremely large (Yang 2008). In their essence, meta-heuristics incorporate randomness and a local search in their process (ye 2017). These features support meta-heuristics to find a suboptimal solution when applying traditional algorithms for evaluating every possible solution is impossible. In general, nature-inspired algorithms can be classified into two main categories: Evolutionary Algorithms and Swarm Intelligence algorithms. Evolutionary algorithms are mainly inspired by the Darwinian theory of evolution and natural selection. This research will talk about nine techniques of nature-inspired algorithms to solve the problem of high fuel cost in the smart grid, and shows the results of implement three units' system, six-unit system, and IEEE thirty bus on nine algorithms to reach the general aim of the proposed research.

1.2 Problem Statement

The mathematical principle of the ELD problem depends on formulating the power cost as a minimization of an optimization function. The primary goal of the ELD problem is to decrease the generation cost of power distribution and the allocated power network not reliable and cannot fulfill the customer's needs and minimize power losses during transmission.

1.3 Proposed Solution

The proposed solution for the ED Problem by using the mathematical formulating of the power cost to a provided the minimization of an optimization function by testing various (nine) algorithms to known the convergence performance of proposed search algorithms to get the best algorithms.

1.4 Objectives

The objectives of this study are to decrease the generation cost of power distribution. For a particular thermal system that consists of n generators, the total generation cost use meta-heuristic search algorithms adopted in this study shall be used to optimize the cost function of the generated power. This function can be presented as given in Equation to testing nine algorithms. This was reported in terms of the total cost, time, and load fulfillment accuracy.

1.5 Methodology

This study shall be used MATLAB to execute optimize the cost function of the generated power to solve multi-objective optimization procedures using LP. The LP technique with piecewise linearization provided an overall economic benefit.

1.6 Scope

The scope of this Study for using Optimization Techniques to Minimizing Fuel Cost in the Smart Grid.

1.7 Thesis Outlines

The rest of this thesis prospectus is structured as follows:
Chapter 2: Review of Relevant Literature. This chapter describes scholarly articles and journals that explain a variety of concepts that are Meta-heuristics Algorithms.
Chapter 3: Problem Formulation. This chapter emphasizes the explains the minimize fuel cost formulating to economic dispatch problem.
Chapter 4: Nature-inspired Meta-heuristics Search Algorithms. This section describes scholarly articles and journals that explain a variety of concepts that are Meta-heuristics Algorithms.
Chapter 5: Experimental Results. This chapter shows the results of implement three units system, six unit system, and IEEE thirty bus on nine algorithms to reach the general aim of the proposed research.
Chapter 6: Conclusions and Future work. This chapter shows the conclusion and Future work of this research.
Chapter 7: Bibliography. A list of references

Chapter 2

Literature Review

Meta-heuristics are widely known as efficient approaches for many hard optimization problems. The classification of meta-heuristic divided to single solution and population. We determine optimization problems as problems that cannot be solved to optimal, by any exact method within a reasonable time limit.

Can be divide problems into several categories depending on whether they are continuous or discrete, constrained or unconstrained, mono or multi-objective, static or dynamic. Meta-heuristics can be used to solve all these problems. Meta-heuristics define an algorithm designed to solve a wide range of hard optimization problems without having to deeply adapt to each problem.

Indeed, the Greek prefix" meta" is used to indicate that these algorithms are "higher-level" heuristics, in contrast with problem-specific heuristics. Meta-heuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm to solve them.

One pioneer contribution is the proposition of the simulated annealing method by (S. Kirkpatrick 1983). In 1986, the tabu search was proposed by (Glover 2008), and the artificial immune system was proposed by (J.D. Farmer 1986). In 1988, Koza registered his first patent on genetic programming, later published in 1992(Koza 1992). In 1989, Goldberg published a well-known book on genetic algorithms (Goldberg 1989). In 1992, Dorigo completed his PhD thesis,in which he describes his innovative work on ant colony optimization(Dorigo 1992).In 1993, the first algorithm based on bee colonies was proposed by Walker et al (A. Walker 1993).Another significant progress is the development of the particle swarm optimization by Kennedy and Eberhart in 1995 (J. Kennedy 1995).The same year, Hansen and Ostermeier proposed CMAES(N. Hansen 1995).In 1996, Mu¨hlenbeinand Paaß proposed the estimation of distribution algorithm(Mu¨hlenbein 1996). In 1997, Storn and Price proposed differential evolution (R.M. Storn 1997). In 2002, Passino introduced an optimization algorithm based on bacterial foraging (Passino 2002).Then, Simon proposed a bio-geographybased optimization algorithm in 2008(Simon 2008).

2.1 Single-solution based metaheuristics

Called trajectory methods are Reverse from population-based metaheuristics, they start with a single initial solution and move away from it, describing a trajectory in the search space. Trajectory methods mainly encompass the simulated annealing method, the tabu search, the GRASP method, the variable neighborhood search, the guided search, the iterated local search, and their variants.

2.1.1 Simulated Annealing

The origins of the Simulated Annealing method (SA) are in statistical mechanics (Metropolis algorithm (N. Metropolis 1953). It was first proposed by Kirkpatrick et al. (S. Kirkpatrick 1983), and independently by Cerny (Cerny 1985).

SA is inspired by the annealing technique used by the metallurgist s to obtain a" well ordered" solid state of minimal energy (while avoiding the" meta-stable" structures, characteristic of the local minimum of energy). This technique consists of carrying a material at a high temperature, then in lowering this temperature slowly.

SA transposes the process of the annealing to the solution of an optimization problem: the objective function of the problem, similar to the energy of a material, and then minimized, by introducing a fictitious temperature T, which is a simple controllable parameter of the algorithm.

SA has been successfully applied to several discrete or continuous optimization problems, though it has been found too avid or unable to solve some combinatorial problems. The adaptation of SA to continuous optimization problems has been particularly studied in a wide bibliography can be found in (Alba 2005) (H.G. Beyer 2002) (N.E. Collins 1988) (Fleischer 1995) (C. Koulamas 1994) (P.V. Laarhoven 1987) (Ed 2008).

2.1.2 Tabu search

Tabu Search (TS) was formalized in 1986 by Glover (Glover 2008). TS was designed to manage an embedded local search algorithm. It explicitly uses the history of the search, both to escape from local minima and to implement an explorative strategy. Its main characteristic is indeed based on the use of mechanisms inspired by human memory. It takes, from this point of view, a path opposite to that of SA, which does not use memory, and thus is unable to learn from the past.

2.2 Population-based meta-heuristics

Population-based meta-heuristics deal with a set (that means a population) of solutions rather than with a single solution. The most studied population-based methods are related to Evolutionary Computation (EC) and Swarm Intelligence (SI). EC algorithms are inspired by Darwin's evolutionary theory, where a population of individuals is modified through recombination and mutation operators. In SI, the idea is to produce computational intelligence by exploiting simple analogs of social interaction, rather than purely individual cognitive abilities.

2.2.1 Evolutionary Computation

Evolutionary Computation (EC) is the general term for several optimization algorithms that are inspired by the Darwinian principles of nature's capability to evolve living beings well adapted to their environment. Usually found grouped under the term of EC algorithms (also called Evolutionary Algorithms (EAs)), are the domains of genetic algorithms (Holland 1975a), evolution strategies (Rechenberg 1973), evolutionary programming (L.J. Fogel 1966), and genetic programming (Koza, Bennett, Andre, Keane, and Dunlap 1997). Despite the differences between these techniques, which will be shown later, they all share a common underlying idea of simulating the evolution of individual structures via processes of selection, recombination, and mutation reproduction, thereby producing better solutions.

2.2.2 Genetic algorithm

The Genetic Algorithm (GA) is the most well-known and most used evolutionary computation technique. It was originally developed in the early 1970s at the University of Michigan by John Holland and his students, whose research interests were devoted to the study of adaptive systems (Holland 1975b). The basic GA is very generic, and there are many aspects that can be implemented differently according to the problem: representation of solution (chromosomes), selection strategy, type of crossover and mutation operators, etc. The most common representation of the chromosomes applied in GAs is a fixed-length binary string. Simple bit manipulation operations allow the implementation of crossover and mutation operations.

These genetic operators form the essential part of the GA as a problem-solving strategy. Emphasis is mainly concentrated on the crossover as the main variation operator, which combines multiple (usually two) individuals that have been selected together by exchanging some of their parts. There are various strategies to do this, e.g., n-point and uniform crossover. An exogenous parameter personal computer (crossover rate) indicates the probability per individual to undergo crossover. Typical values for personal computer are in the range [0.6,1.0] (B¨ack and Schwefel 1993).

Individuals for producing offspring are chosen using a selection strategy after evaluating the fitness value of each individual in the selection pool. Some of the popular selection schemes are roulette-wheel selection, tournament selection, ranking selection, etc. A comparison of selection schemes used in GAs is given in (T. Blickle 1995) (D.E. Goldberg 1991). After crossover, individuals are subjected to mutation. Mutation introduces some randomness into the search to prevent the optimization process from getting trapped into local optima. It is usually considered as a secondary genetic operator that performs a slight perturbation to the resulting solutions with some low probability.

Typically, the mutation rate is applied with less than one percent probability, but the appropriate value of the mutation rate for a given optimization problem is an open research issue. The replacement (survivor selection) uses the fitness value to identify the individuals to maintain as parents for successive generations and is responsible to assure the survival of the fittest individuals. Interested readers may consult the book by Goldberg (Goldberg 1989) for more detailed background information on GAs.

2.2.3 Differential evolution

The Differential Evolution (DE) algorithm is one of the most popular algorithms for continuous global optimization problems. It was proposed by Storn and Price in the ’90s (R.M. Storn 1997) in order to solve the polynomial fitting problem and has proven to be a very reliable optimization strategy for many different tasks. Like any evolutionary algorithm, a population of candidate solutions for the optimization task to be solved is arbitrarily initialized. For each generation of the evolution process, new individuals are created by applying reproduction operators (crossover and mutation). The fitness of the resulting solutions is evaluated and each individual (target individual) of the population competes against a new individual (trial individual) to determine which one will be maintained into the next generation.

The trial individual is created by recombining the target individual with another individual created by mutation (called mutant individual). Different variants of DE have been suggested by Price et al. (K.V. Price 2005) and are conventionally named DE/ x/ y/ z, where DE stands for Differential Evolution, x represents a string that denotes the base vector, i.e. the vector being perturbed, whether it is “ rand” (a randomly selected population vector) or “ best” (the best vector in the population with respect to fitness value), y is the number of difference vectors considered for perturbation of the base vector x and z denotes the crossover the scheme, which may be binomial or exponential. The DE/rand/1/bi n-variant, also known as the classical version of DE, is used later on for the description of the DE algorithm.

2.2.4 Swarm intelligence

The Swarm Intelligence (SI) is an innovative distributed intelligent paradigm for solving optimization problems that takes inspiration from the collective behavior of a group of social insect colonies and of other animal societies. SI systems are typically made up of a population of simple agents (an entity capable of performing/executing certain operations) interacting locally with one another and with their environment. These entities with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global and self-organized behavior. Several optimization algorithms inspired by the metaphor s of swarming behavior in nature are proposed. Ant colony optimization, Particle Swarm

Optimization, Bacterial foraging optimization, Bee Colony Optimization, Artificial Immune Systems, and Bio-geography-Based Optimization.

Fundamentals of Computational Swarm Intelligence Book (Engelbrecht 2006) introduces the reader to the mathematical models of social insects' collective behavior and shows how they can be used in solving optimization problems. Another book by Chan et al. (F.T.S. Chan 2007) aims at presenting recent developments and applications concerning optimization with SI, making a focus on Ant and Particle Swarm Optimization. (S. Das 2008) provide a detailed survey of the state-of-the-art research centered around the applications of SI algorithms in bioinformatics. (A. Abraham 2008) deals with the application of SI in data mining.

Particle swarm optimization

The Particle Swarm Optimization (PSO) was initially introduced in 1995 by James Kennedy and Russell Eberhart as a global optimization technique (J. Kennedy 1995). It uses the metaphor of the flocking behavior of birds to solve optimization problems. There are a number of differences between PSO and evolutionary optimization illustrated in (Angeline 1998), where some of the philosophical and performance differences are explored.

In the PSO algorithm, many autonomous entities (particles) are stochastically generated in the search space. Each particle is a candidate solution to the problem, and is represented by a velocity, a location in the search space, and has a memory that helps it in remembering its previous best position. A swarm consists of N particles flying around in a D-dimensional search space. Moreover, every particle swarm has some sort of topology describing the interconnections among the particles. The set of particles to which a particle i is topologically connected is called i's neighborhood. The neighborhood may be the entire population or some subset of it. Various topologies have been used to identify "some other particle" to influence the individual. The two most commonly used ones are known as gbest (for "global best") and lbest (for "local best"). The traditional particle swarm topology known as gbest was one where the best neighbor in the entire population influenced the target particle.

While this may be conceptualized as a fully connected graph. The lbest topology, introduce d in (R.C. Eberhart 1995), is a simple ring lattice where each individual is connected to K = 2 adjacent members in the population array, with toroidal wrapping (naturally, this can be generalized to K ¿ 2) (J. Kennedy 1995). pointed out that the gbest topology had a tendency to converge very quickly with a higher chance of getting stuck in local optima. On the other hand, the lbest topology was slower but explored more fully, and typically ended up at a better optimum (J. Kennedy 2002).

Chapter 3

Economic Dispatch of Thermal Units

The complication of interconnections and the scope of the areas of electric power systems that are controlled in a synchronized way is fast increasing. This requires optimal allocation of the outputs of a great number of active generators. Whether a generator should participate in sharing the load at a given break of time is a problem of unit commitment. when the unit commitment problem has been solved, it becomes a problem of optimal allocation of the available generations to meet the predicted load demand for the current interval. At a current-day energy management center, highly developed optimization techniques are used to govern not only the optimal outputs of the active generators, but also the optimal settings of various control devices such as the tap settings of load tap changers (LTCs), outputs of VAR compensating devices, desired settings of phase convert etc.

The favorite objective for such optimization problems can be many, such as the minimization of the cost of generation, minimization of the total power loss in the system, minimization of the voltage deviations, and maximization of the reliability of the power supplied to the customers. One or more of these objectives can be considered while formulating the optimization strategy. Determination of the real power outputs of the generators so that the total cost of generation in the system is minimized is traditionally known as the problem of economic load dispatch (ELD). popular of generating systems are of three types: nuclear, hydro, and thermal (using fossil fuels such as coal, oil and gas). Nuclear plants tend to be operated at constant output power levels. Operating cost of hydro units do not change much with the output. The operating cost of thermal plants, however, change great with the output power level. In this chapter, we will discuss the problem of ELD for power systems consisting of thermal units only as generators.

3.1 Economic Dispatch Problem

Primary we formulate the ELD problem neglecting transmission losses. This is justified when a group of generators are connected to a particular bus-bar, as in the case of individual generating units in a power plant, or when they are physically located very close to each other. This ensures that the transmission losses can be neglected due to the short distance involved. One such system configuration is shown in Figure 3.1, where N thermal units are connected to a single bus-bar that is supplying a load Pload. Input to each unit is expressed in terms of cost rate (say \$/h).

The total cost rate is the sum of cost rates of individual units. The essential operating constraint is that the sum of the power outputs must be equal to the load (note that we are neglecting power losses here).

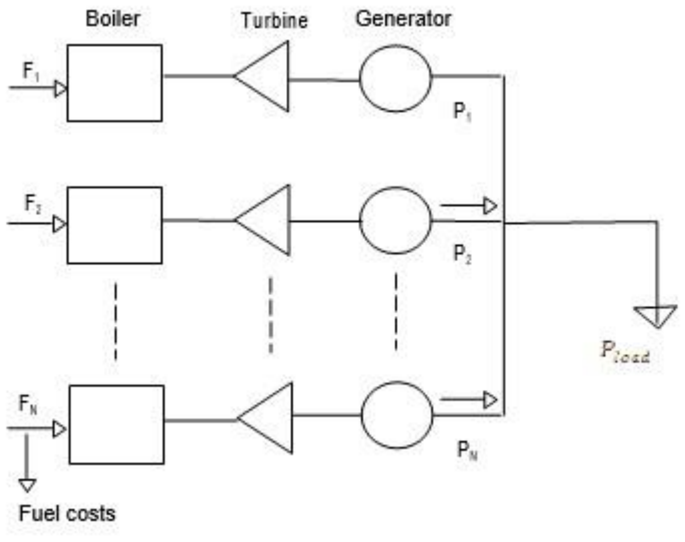

Figure 3.1: N thermal units connected to a bus to serve a load P_{load}

3.1.1 Fuel Cost Characteristics

The economic dispatch problem is the determination of generation levels such that the total cost of generation becomes minimum for a defined level of load. Now, for thermal generating units, the cost of fuel per unit power output varies significantly with the power output of the unit. So, one needs to consider the fuel cost characteristics of the generators while finding their optimal real power outputs. The fuel cost characteristics is shown below Figure.

Mostly, the cost of work, supply and maintenance are fixed. Pmin is the output level below which it is uneconomical or technically infeasible to operate the units. Pmax is the maximum output power limit. For formulating the dispatch problem, fuel costs are usually represented as a quadratic function of output power, as shown below.

$$F(P) = aP^2 + bP + c \qquad (3.1)$$

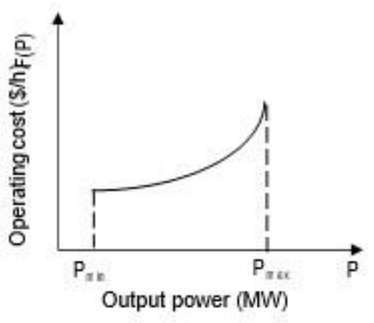

Figure 3.2: Typical fuel cost characteristics

3.1.2 Problem Formulation

The total fuel cost of operating N generators is given by

$$F_T = F_1(P_1) + F_2(P_2) + \ldots + F_N(P_N) = \sum_{i=1}^{N} F_i(P_i)$$

Neglecting transmission losses, total generation should meet the total load. Hence, the equality constraint is,

$$\sum_{i=1}^{n} P_i = P_{Load} \tag{3.2}$$

Based on the maximum and minimum power limits of the generators, following inequality constraints can be imposed:

$$P_{i,min} \leq P_i \leq P_{i,max} \tag{3.3}$$

This is a constrained optimization problem that can be solved by multiple method. Economic Operation of Power Systems (A. J. Wood 2006) (Kirchmayer 1979)

• Swarm-based algorithms such as particle swarm optimization (Gaing 2003; Kuo 2008; Rahmani, Othman, Yusof, and Khalid 2012; Dewangan, Jain, and Huddar 2010), cuckoo search (Nguyen and Vo 2015; Sen and Acharjee 2016), ant colony optimization (Aristidis 2006),

- Evolutionary-based algorithms such as evolutionary algorithms (Sahoo, Dash, Prusty, and Barisal 2015), genetic algorithm (Gaing 2003; Chen and Chang 1995), harmony search algorithm (Chakraborty, Roy, Panigrahi, Bansal, and Mohapatra 2012), biogeography-based optimization (Bhattacharya and Chattopadhyay 2010), and • Trajectory-based algorithms such as simulated annealing (Bhattacharya and Chattopadhyay 2011).

3.2 Mathematical Formulation of ELD

In this section, we start by providing a formulation to the ELD problem. The economic dispatch problem objective is to maximize the economic welfare of a power network under various operation constraints. Assume we have a network with *n* buses (nodes). The unconstrained ELD problem can be formulated as:

$$\begin{aligned} Min\ C_k(P_k) &= C_1(P_1) + \cdots + C_1(P_n) \\ &= \sum_{k=1}^{n} C_k(P_k) \end{aligned} \tag{3.4}$$

where I_k represents the net power injection at bus k, and $C_k(P_k)$ is the cost function of producing power at bus k. A power system with this given configuration can be presented as in Figure 3.3 where n thermal units are connected to a single bus-bar that is supplying a load power P_k. The input to each unit is expressed in terms of cost rate (say \$/h) P_k. $k = 1,...,n$, n is the number of power generator units. The cost presented in Equation 3.4 can be approximated in a quadratic form as given in Equation 3.5 for minimization purposes (Bergen 1986; Wood and Wollenberg 2010).

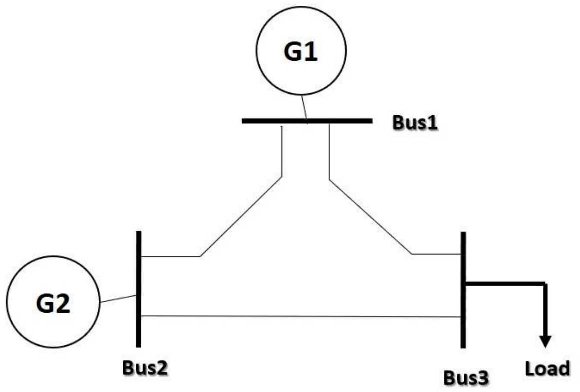

Figure 3.3: Two generators and Three Bus power system

$$Min\ C_k(P_k) = \sum_{k=1}^{n} \alpha_k P_k^2 + \beta_k P_k + \gamma_k \tag{3.5}$$

P_k is the generated power from generator unit k, α_k,β_k and γ_k are the fuel cost coefficients of unit i. Two types of constraints shall be considered while solving this problem; equality constraints and inequality constraints.

3.2.1 Power Balance Equality Constraints

A real power system has to generate enough power such that it covers both the demand and the transmission lines power loss. It is known that the power produced at any power station go through large and complex networks such as transformers, transmission lines, cables and additional equipment to supply the end users of their demand. Therefore, it is always the case that the power units in a network always produce extra power not only to match the demand but also to recover the waste of transmission power. This difference in the generated and distributed power P_G is recognized as Transmission and Distribution loss power P_L. Any lack in the generated power P_G will cause shortage in feeding the power in demand P_D which could be a reason for several problems for the system and loads (See Equation 3.6).

$$P_G = \sum_{k=1}^{n} P_k = P_D + P_L \tag{3.6}$$

where P_D is the load demand and P_L is the transmission lines loss, while n and P_k have the same definition as in Equation 3.5.

To consider the effect of transmission losses in our cost computation, we adopted the loss coefficient method which proposed by *Kron and Kirchmayer* (Bergen 1986; Wood and Wollenberg 2010). In this method, a matrix ζ is defined as ”the transmission loss coefficients matrix” used to include the power loss. ζ is a square matrix with a dimension of $R^{n\times n}$ while n is the number of power generation units in the system. Equation 3.7 describes the definition of P_L based the transmission loss ζ-matrix.

$$P_L = \mathrm{XX} P_i \zeta_{ij} P_j \quad \begin{smallmatrix} n & n \\ i=1 & j=1 \end{smallmatrix} \tag{3.7}$$

where P_{Loss} is the transmission power loss, P_i,P_j are the power generated from any two power generator units i,j. Meanwhile, ζ_{ij} is the elements of the matrix ζ between i and j power generator units.

3.2.2 Generation Limit Inequality Constraints

The generated power from the power generation system should satisfy number of constraints based on the capacity of the generation unit. For instance, the generation units cannot exceed a certain power generation unit since this nay cause instability for the synchronous generators. Meanwhile, generating less power than a minimum limit may cause the rotor to over speed. These limitations for the *k*th generator are described in Equation 3.9 and presented in (Saadat 2008).

$$P_k^{min} \leq P_k \leq P_k^{max}, k = 1, \ldots, n \tag{3.8}$$

where P_k^{min} and P_k^{max} are the limitation of generation for the *k*th generation unit.

3.2.3 Generation Limit Inequality Constraints

The generated power from the power generation system should satisfy number of constraints based on the capacity of the generation unit. For instance, the generation units cannot exceed a certain power generation unit since this nay cause instability for the synchronous generators. Meanwhile, generating less power than a minimum limit may cause the rotor to over speed. These limitations for the *k*th generator are described in Equation 3.9 and presented in (Saadat 2008).

$$P_k^{min} \leq P_k \leq P_k^{max}, k = 1,\ldots,n \tag{3.9}$$

whereP_k^{min} andP_k^{max} are the limitation of generation for the *k*th generation unit.

Chapter 4

Nature-inspired Meta-heuristics Search Algorithms

Meta-heuristics are global search algorithms and their goal is to find an acceptable solution within a reasonable time frame when the problem is very complex and the search space is extremely large. In their essence, meta-heuristics incorporate randomness and a local search in their process. These features support meta-heuristics to find a suboptimal solution when applying traditional algorithms for evaluating every possible solution is impossible. However, this does not grantee that meta-heuristics will always find the optimal solution neither that they will work. Basically, there are two main components of the meta-heuristics algorithms: exploration and exploitation. In the exploration component, the algorithm tries to explore and test different areas in the search space, while on the other hand, in the exploitation component, the algorithm tries to focus the search around some suboptimal found solutions (Yang 2008).

Most of nature-inspired algorithms are population-based algorithms where they start by randomly generating a predetermined number of candidate solutions (also called individuals) then they start to iteratively update the generated solutions using a specific designed mechanism. In every iteration, the algorithm evaluates all individuals using a fitness function to assess their quality considering them as possible solutions for the targeted problem. In some meta-heuristics, fitness values affect the search direction of the algorithm.

In general, nature-inspired algorithms can be classified into two main categories: Evolutionary Algorithms and Swarm Intelligence algorithms. Evolutionary algorithms are mainly inspired by the Darwinian theory on evolution and natural selection. Best example of this type is the well-regarded Genetic Algorithm (GA). GA was first proposed and designed in the works of John Holland (Holland 1992). GA is distinguished by its reproduction operators namely; the crossover and mutation operators. On the other side, most of the Swarm Intelligence algorithms are inspired by the movement or interaction of some families of birds, fish or animals in nature. A well-regarded example of this category is the Particle Swarm Optimization (PSO). PSO was first introduced by Kennedy and Eberhart in 1995 (J. Kennedy 1995; Poli, Kennedy, and Blackwell 2007). In PSO, individuals (or particles) are updated based on the best-found solution by all individuals and the best solution found by the updated individual itself.

4.1 Genetic Algorithm

GA is an evolutionary approach, which applies evolutionary operators and a population of solutions to achieve a global optimal solution. Gas includes selection, recombination and mutation. Candidate solutions to the problem are encoded as chromosomes, and then a fitness function inversely proportional to the mean squared error value is applied to determine the chromosomes surviving likelihood in the next generation. In GA we use a model of the natural selection in real life, where an initial population of solutions called individuals is randomly generated. The algorithm produces new solutions of the population by genetic operations, such as reproduction, crossover and mutation (**?**). The new generation consists of the possible survivors with the highest fitness score, and new individuals estimated from the previous population using the genetic operations.

GA search the solution space of a function through the use of simulated evolution, i.e., the survival of the fittest strategy (**?**). GAs was used to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population. which are individual solutions (analogous to chromosomes) of the state space. These operators, which rely on probability rules, are applied to the population, and successive generations are produced.

In general, the starting search for an optimal solution begins with a randomly generated population of chromosomes. Each generation will have a new set of chromosomes obtained from the application of the operators. A fitness, or objective function, is defined according to the problem. The parent selection process ensures that the fittest members of the population have highest probability of becoming parents, in the hope that their offspring will combine desirable features, and have superior fitness, to both. The algorithm terminates either when a set of generation number is reached, or the fitness has reached a" satisfactory" level. The use of a GA requires the determination of **six fundamental issues:**

1. Representation
2. Distribution of initial population
3. Fitness Function
4. Selection Mechanism
5. Reproduction Parameters (i.e., Crossover and Mutation)
6. Termination Criteria

The main steps for GA procedure can be summarized as follows:

1. Generate an initial population.
2. Evaluate the fitness of each individual according to the given fitness function.
3. Select the fittest individual for mating.
4. Apply reproductive operators (e.g., crossover, mutation) to create offspring.
5. Evaluate the fitness of the offspring and select the fit individuals from the current generation and the offspring. They form the population of the next generation.
6. Stop if stopping criterion is met, else go to step 3.

The GA can be presented as in Figure 4.1.

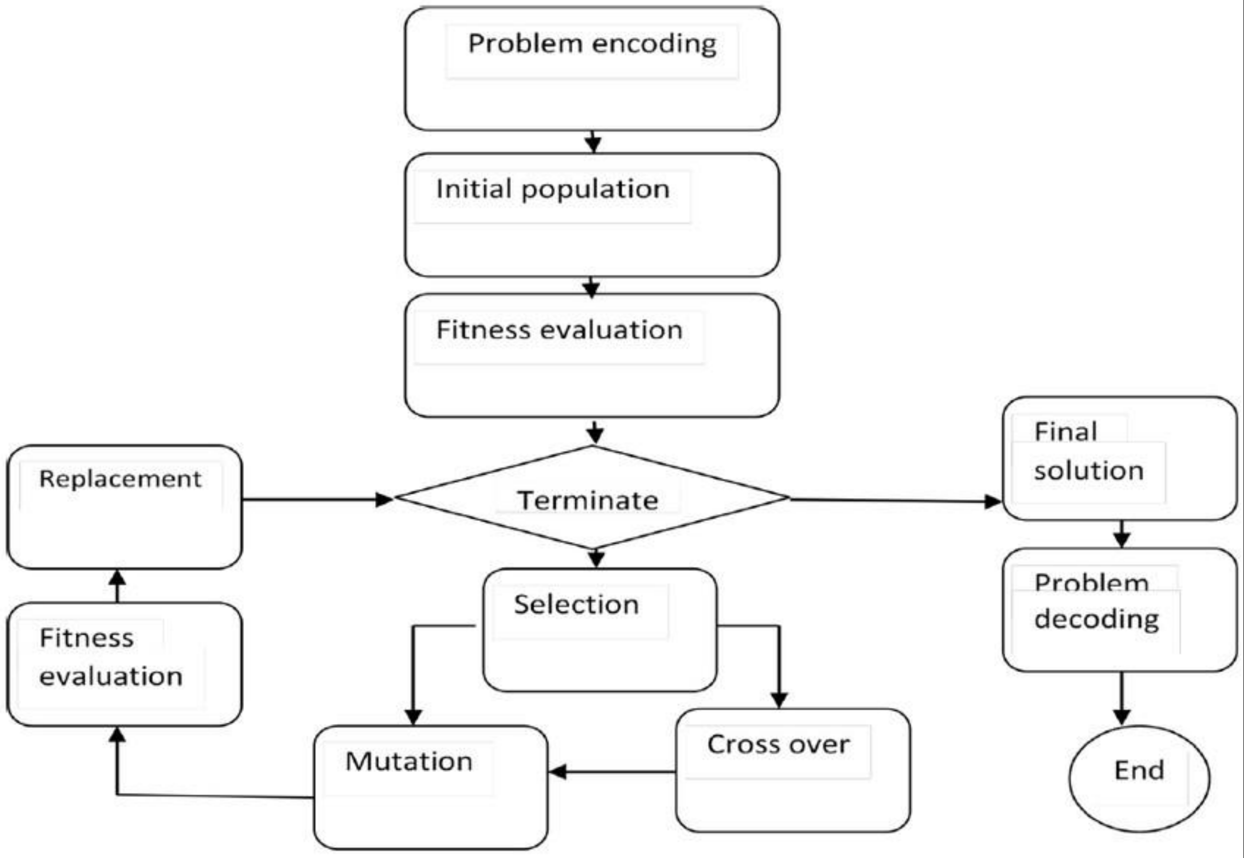

Figure 4.1: Flowchart for GAs

4.2 Particle Swarm Optimization

PSO belongs to a class of swarm intelligence techniques that are used to solve optimization problems (J. Kennedy 1995). Each particle in PSO is updated by following two" best" values:

- *pbest*: Each particle keeps track of its coordinates in the solution space which are associated with the best solution (i.e fitness) that has achieved so far by that particle. This value is called personal best, *pbest*.
- *gbest*: It is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called Global Best, *gbest*.

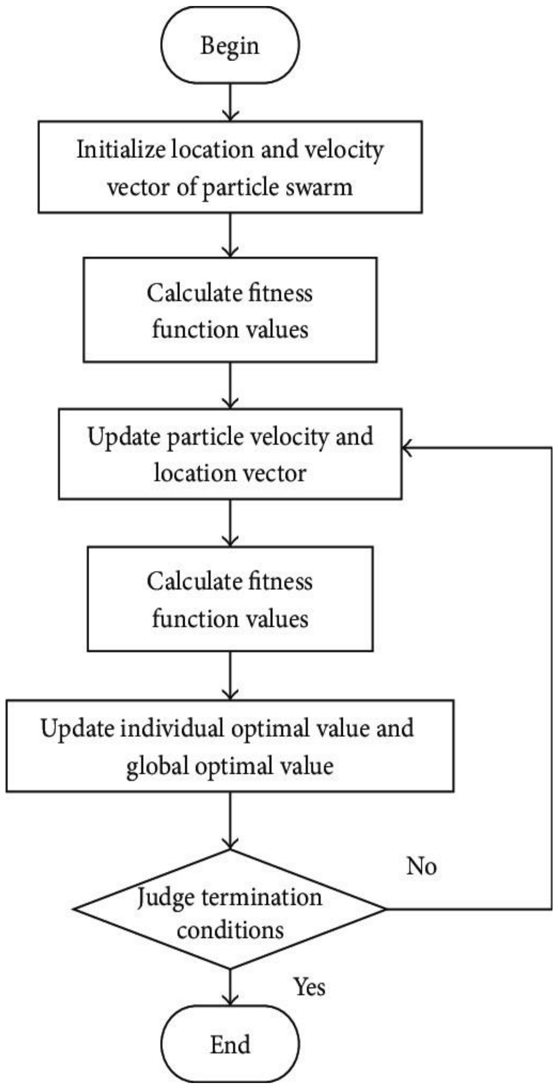
Begin
Initialize location and velocity vector of particle swarm
Calculate fitness function values
Update particle velocity and location vector
Calculate fitness function values
Update individual optimal value and global optimal value
Judge termination conditions
No
Yes
End

Figure 4.2: Flowchart for PSO

The PSO Algorithm works as follows:

Let X and V denote the particles position and its corresponding velocity in search space respectively. At iteration K, each particle i has its position defined by X_i^K = $[X_{i,1}, X_{i,2}, ..., X_{i,N}]$ and a velocity is defined as $V_i^K = [V_{i,1}, V_{i,2}, ..., V_{i,N}]$ in search space N. Velocity and position of each particle in the next iteration can be calculated as:

$$V_{i,nk+1} = W * V_{i,nk} + C_1 * rand_1 * (pbest_{i,n} - X_{i,nk}) + C_2 * rand_2 * (gbest_n - X_{i,nk}) \quad (4.1)$$

$$where i = 1,2.....m \qquad and \qquad n = 1,2,...,N$$

$$X_{i,nk+1} = \text{□□□} X X_{min,i,nk} + V_{,i,nk+1}, if X if X_{ikmin,i,n+1} < X <_{min,i,n} = X_{ik+1} <= X_{max,i,n} \quad (4.2)$$

$$\text{□□} X_{max,i,n}, \qquad if X_{ik+1} > X_{max,i,n}$$

Algorithm for PSO initialize each particle to contain N_c randomly selected cluster means. t=1 to t_{max}(maximum number of iterations) each particle i each pixel Z_p calculate $d(Z_p, m_{ij}) for all clusters c_{ij}$ Assign $Z_p to C_{ij}$ where $d(Z_p, m_{ij}) = min_{\forall c=1,...,N_c} d(Z_p, m_{ic})$ $d(Z_p, m_{ij})$ represents the euclidean distance between the p-th pixel Z_p and the centroid of j-th cluster of pixel i Calculate the fitness function f(xi(t),Z) where Z is a matrix representing the assignment of pixels to clusters of particle i Update the personal best and the global best positions Update the cluster centroids

The inertia weight W is an important factor for the PSO convergence. It is used to control the impact of previous history of velocities on the current velocity. A large inertia weight factor facilitates global exploration (i.e., searching of new area) while small weight factor facilitates local exploration. Therefore, it is better to choose large weight factor for initial iterations and gradually reduce weight factor in successive iterations. This can be done by using

$$W = W_{max} - (W_{max} - W_{min}) \times \frac{Iter}{Iter_{max}} \quad (4.3)$$

where max and W_{min} are initial and final weight respectively, *Iter* is current iteration number and $Iter_{max}$ is maximum iteration number.

Acceleration constant C_1 called cognitive parameter pulls each particle towards local best position whereas constant C_2 called social parameter pulls the particle towards global best position (ye 2017). The particle position is modified by Equation (4.2). The process is repeated until stop-ping criterion is reached.

The number of threshold levels is the dimension of the problem. For example, if there are ”m” threshold levels, the i^{th} particle is represented as follows:

$$X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,m}) \quad (4.4)$$

Its implementation consists of the following steps:

- Initialization of the swarm: For a population size p, the particles are randomly generated between the minimum and the maximum limits of the threshold values.
- Evaluation of the objective function: The objective function values of the particles are evaluated using the objective functions.
- Initialization of *pbest* and *gbest*: The objective values obtained above for the initial particles of the swarm are set as the initial *pbest* values of the particles. The best value among all the *pbest* values is identified as *gbest*.
- Evaluation of velocity: The new velocity for each particle is computed using the Equation.
- Update the swarm: The particle position is up-dated using Equation 4.1. The values of the objective function are calculated for the updated positions of the particles. If the new value is better than the previous *pbest*, the new value is set to *pbest*. Similarly, *gbest* value is also updated as the best *pbest*. • Stopping criteria: If the stopping criteria are met, the positions of particles represented by *gbest* are the optimal threshold values. Otherwise, the procedure is repeated from step 4.

There are many advantages of PSO. They include:

1. PSO is easy to implement and only few parameters have to be adjusted.
2. Unlike the GA, PSO has no evolution operators such as crossover and mutation.
3. In GAs, chromosomes share information so that the whole population moves like one group, but in PSO, only global best particle (*gbest*) gives out information to the others. It is more robust than GAs.
4. PSO can be more efficient than GAs; that is, PSO often finds the solution with fewer objective function evaluations than that required by GAs (P. R. Lorenzo 2017).

5. Unlike GAs and other heuristic algorithms, PSO has the flexibility to control the balance between global and local exploration of the search space.

4.3 Crow Search Algorithm

The CSA is a recent nature-inspired metaheuristic which was proposed in (Askarzadeh 2016). The abstract ideas of CSA are inspired by the behavior of crows birds in nature. Crows are considered as one of the most intelligent birds. It was reported in different studies that crows show clever behaviors such as the ability to hide their exceeded food and the ability to find it again. Moreover, crows communicate in sophisticated way and they have good memory to recognize objects. As a search algorithm, CSA was implemented based on the following four main points:

- Crows exist in nature as flocks so the CSA is formed as a population-based algorithm.
- Crows can remember the place where they hide their food and retrieve it again.
- Crows can watch other animal to steal their food.
- Crows manage to protect their store of food with a ratio.

In CSA, a solution for a targeted problem is represented as the position of the crow at a given time as shown in Equation 4.5, where $x_{i,G}$ is the position of crow *i* at iteration *G*. Note that we used *G* to denote the concept of *iteration* which is analogous to *generation* in GA or DE. $x_{i,G}$ is consisted of *D* variables which represents the dimension of the problem.

$$x_{i,G} = \left[x_1^{i,G}, x_2^{i,G}, \ldots, x_D^{i,G}\right] \tag{4.5}$$

CSA is a population-based metaheuristic. As most of optimizers that belong to this family, it starts by randomly generating a group of possible solutions of size *N* called *flock of crows*. Therefore, the size of the population is *N*×*D*. CSA incorporates also the concept of memory, which represents the qualities of the positions of the crows. The quality of each position is measured by the fitness function and stored in an array as given in Equation 4.6.

$$Memory = \begin{bmatrix} m_1^1 & m_2^1 & \dots & m_d^1 \\ m_1^2 & m_2^2 & \dots & m_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^N & m_2^N & \dots & m_d^N \end{bmatrix} \tag{4.6}$$

Crows in the flock update their position using the following mechanism: each crow *i* selects another crow *j* from the flock to follow in a hope to find the food hidden by the latter crow m^j. This movement is represented as given in Equation 4.7.

$$x^{i,G+1} = \begin{cases} x^{i,G} + r_i \times fl^{i,G} \times \left(m^{j,G} - x^{i,G}\right) & r_j \geq AP^{j,G} \\ a\,random\,position & otherwise \end{cases} \tag{4.7}$$

where r_j is a random generated number drawn from uniform distribution between 0 and 1, while $AP^{j,G}$ is the awareness probability of crow *j* at iteration *G*.

The fitness of the new position is checked, if its quality is better than the current one then the position is updated. Otherwise, the crow stays in the same position. Then, the memory can be updated as given in Equation 4.8.

$$m^{i,G+1} = \begin{cases} x^{i,G+1} & f\left(x^{i,G+1}\right) is\,better\,than\,f\left(m^{i,G}\right) \\ m^{i,G} & Otherwise \end{cases} \tag{4.8}$$

where *f* denotes the value of fitness function. Similarly, if the fitness value of the new position is better than the memorized position then the crow updates its memory accordingly. The processes of generating new positions, evaluating them and updating memories are repeated until a predefined termination condition is met.

4.4 Differential Evolution

Differential Evolution (DE) is one of the most well-regarded evolutionary algorithms (R.M. Storn 1997). Similarity to other evolutionary algorithms, DE first initializes the first population. It then performs difference-vector based mutation, crossover, and selection. During the optimization process each solution is evaluated by an objective function and assigned an objective value. Each of these steps are discussed as follows (Das and Suganthan 2011).

4.4.1 Initialization

DE is considered as a real-parameter optimization algorithm. Therefore, each variable has a minimum and maximum. The following vector is initialized considering the lower and upper bounds of each variable:

$$\sim x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}] \tag{4.9}$$

where *i* is an index to refer to *i*th vector in the population, *G* is the generation number, and *D* indicates the number of dimension (variables of the problem).

The initialization is done using the following equation:

$$x_{j,i,0} = ub_j + r \cdot (ub_j - lb_j) \tag{4.10}$$

where ub_j is the upper bounds in the *j*th dimension and lb_j shows the lower bout in the *j*th dimension.

4.4.2 Mutation

Mutation in nature occurs in genome with random changes in the genes. In DE, if a solution faces mutation, it is called *donor*. To perform mutation in DE, three vectors are sampled randomly: $\vec{X}_{r_1^i}, \vec{X}_{r_2^i}, \vec{X}_{r_3^i}$. This means the indices of these three vectors are randomly chosen between one and the maximum number of vectors in the population. To perform mutation, the difference between two of these vectors are calculated (and normalized) for each donor vector. Their difference are then added up to the third vector, which give the final donor vector. These steps can be represented in an equation as follows:

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot \left(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}\right) \tag{4.11}$$

where *F* is a scalar number and normally chosen in the interval of [0.4,1].

4.4.3 Crossover (reproduction)

Crossover is the main operator to promote exploration in any evolutionary algorithm. In DE, crossover is done after mutation. The resulting vector in DE is called the *trial* vector. There are different crossover operators in the literature which mainly varies in the crossover point in the vectors. Regardless of the crossover starting point, it can be formulated as follows:

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{for } j = \langle n \rangle_D \ldots \langle n + L - 1 \rangle_D \\ x_{j,i,G} & \forall j \in [1,D] \end{cases} \tag{4.12}$$

where L shows the donor's number of components, n is randomly chosen in the interval of $[1,D]$, and $\langle\rangle_D$ is a modulo function.

4.4.4 Selection

The selection operator eventually selects the best solutions and allow them to move the the next generation. In DE, if a solution becomes better than its parents, it is replaced by them immediately. Otherwise, the solution is move the the next generation intact. The mathematical formulation for this operator is as follows for minimization problems:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{if } f(\vec{U}_{i,G}) > f(\vec{X}_{i,G}) \end{cases} \tag{4.13}$$

For maximization problems, this equation should be written as follows:

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \geq f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{if } f(\vec{U}_{i,G}) < f(\vec{X}_{i,G}) \end{cases} \tag{4.14}$$

The DE algorithm repeatedly runs these steps until the satisfaction of an end criterion.

4.4.5 Fitness Function

Our objective is to find estimate the optimal power units values $P_k, k = 1,...,n$, n is the number of power units, which minimize the objective criterion L (see Equation 4.15).

$$L = \sum_{k=1}^{n} C_k(P_k) + \lambda \times [\sum_{k=1}^{n} P_k - P_D - P_L] \tag{4.15}$$

where $C_k(P_k)$ is the cost of power generated from generator P_k, P_D is the demand load, P_L is the transmission lost power. λ is an arbitrary chosen parameter with high value to penalize the losses in the cost computation. In our case, λ was set to 100.

4.5 Salp Swarm Algorithm

The Salp Swarm Algorithm (SSA) is a recent nature-inspired optimizer proposed by Mirjalili et al.(Mirjalili 2017). The purpose of SSA is to develop a population-based optimizer by mimicking the swarm behavior of salps in nature (Abbassi 2019).

The performance of the original SSA as an ELM trainer has not been investigated to date. SSA algorithm reveals satisfactory diversification and intensification propensities that make it appealing for evolving ELM training tasks. The unique advantages of SSA cannot be obtained by using some traditional optimizers such as PSO, GWO, and GSA techniques.

The SSA can be considered as a capable, flexible, simple, and easy to be understood and utilized in parallel and serial modes. Furthermore, it has only one adaptively decreasing parameter to make a fine alance between the diversification and intensification inclinations.

In order to avoid immature convergence to local optima (LO), the position vectors of salps are gradually updated considering other salps in a dynamic crowd of agents. The dynamic movements of salps enhance the searching capabilities of the SSA in escaping from LO and immature convergence drawbacks. It also keeps the elite salp found so far to guide other members of swarm towards better areas of the feature space.

The SSA has an iterative generates nature and evolves some random individuals (that means salps) inside the bounding box of the problem. Then, all salps should update their location vectors. The leader salp will attack in the direction of a food source, while all followers can move towards the rest of salps (and leader directly or indirectly) (Mirjalili 2017).

4.6 Harmony Search

4.6.1 Diversification and Intensification

In reviewing other metaheuristic algorithms, we have repetitively focused on two major components: diversification and intensification. They are also referred to as exploration and exploitation (C and A 2003) (M and C 2005). These two components are seemingly dictating each other, but their balanced combination is crucially important to the success of any metaheuristic algorithms (C and A 2003) (M and C 2005).

The best diversification or exploration makes sure the search in the parameter space can explore as many locations and regions as possible in an efficient and effective manner.

It also ensures that the evolving system will not be trapped in biased local optima. Diversification is often represented in the implementation as the randomization and/or additional stochastic component superposed onto the deterministic components. If the diversification is too strong, it may explore more accessible search space in a stochastic manner, and subsequently will slow down the convergence of the algorithm.

If the diversification is too weak, there is a risk that the parameter space explored is so limited and the solutions are biased and trapped in local optima, or even lead to meaningless solutions.

On the other hand, the appropriate intensification or exploitation intends to exploit the history and experience of the search process. It aims to ensure to speed up the convergence, when necessary, by reducing the randomness and limiting diversification.

Intensification is often carried out by using memory such as in Tabu search and/or elitism such as in the genetic algorithms. In other algorithms, it is much more elaborate to use intensification such as the case in simulated annealing and firefly algorithms. If the intensification is too strong, it could result in premature convergence, leading to biased local optima or even meaningless solutions, as the search space is not well explored. If the intensification is too weak, convergence becomes slow.

The optimal balance of diversification and intensification is required, and such a balance itself is an optimization process.

Fine-tuning of parameters is often required to improve the efficiency of the algorithms for a particular problem. A substantial number of studies might be to choose the right algorithms for the right optimization problems (Yang2008), though it lacks systematic guidance for such choices.

4.6.2 Analyze the Harmony Search algorithm

when we analyze the Harmony Search algorithm in the context of the major components of meta-heuristics and try to compare it with other metaheuristic algorithms, we can identify its ways of handling intensification and diversification in the HS method, and probably understand why it is a very successful metaheuristic algorithm.

In the HS algorithm, diversification is essentially controlled by the pitch adjustment and randomization, here there are two subcomponents for diversification, which might be an important factor for the high efficiency of the HS method.

The first subcomponent of composing ‘new music’, or generating new solutions, via randomization would be at least at the same level of efficiency as other algorithms by randomization.

Pitch adjusting is carried out by adjusting the pitch in the given bandwidth by a small random amount relative to the existing pitch or solution from the harmony memory. Essentially, pitch adjusting is a refinement process of local solutions. Both memory considering and pitch adjusting ensure that the good local solutions are retained while the randomization and harmony memory considering will explore the global search space effectively.

The subtlety of this is that it is a controlled diversification around the good solutions (good harmonics and pitches), and it almost acts like an intensification factor as well. The randomization explores the search space more efficiently and effectively; while the pitch adjustment ensures that the newly generated solutions are good enough, or not too far away from existing good solutions.

The intensification is mainly represented in the HS algorithm by the harmony memory accepting rate accept. A high harmony acceptance rate means the good solutions from the history/memory are more likely to be selected or inherited.

Obviously, if the acceptance rate is too low, the solutions will converge more slowly. As mentioned earlier, this intensification is enhanced by the controlled pitch adjustment. Such interactions between various components could be another important factor for the success of the HS algorithm over other algorithms.

In addition, the implementation of HS algorithm is also easier. There is some evidence to suggest that HS is less sensitive to the chosen parameters, which means that we do not have to fine-tune these parameters to get quality solutions.

Furthermore, the HS algorithm is a population-based metaheuristic, which means that multiple harmonics groups can be used in parallel. Proper parallelism usually leads to better implantation with higher efficiency.

The good combination of parallelism with elitism as well as a fine balance of intensification and diversification is the key to the success of the HS algorithm, and in fact, to the success of any metaheuristic algorithms.

These advantages make it very versatile to combine HS with other metaheuristic algorithms such as PSO to produce hybrid meta-heuristics and to apply HS in various applications (Geem ZW and GV 2001) (KS and ZW 2005) (ZW 2006) (M and Mahdavi 2008) (ZW 2008) (ZW 2007).

4.7 Sine Cosine Algorithm

The Sine Cosine Algorithm (SCA)is a novel population-based optimization algorithm for solving optimization problems. The SCA creates multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution using a mathematical model based on sine and cosine functions. Several random and adaptive variables also are integrated to this algorithm to emphasize exploration and exploitation of the search space in different milestones of optimization.

The performance of SCA is benchmarked in three test phases. Firstly, a set of well-known test cases including unimodal, multi-modal, and composite functions are employed to test exploration, exploitation, local optima avoidance, and convergence of SCA.

Secondly, several performance metrics (search history, trajectory, average fitness of solutions, and the best solution during optimization) are used to quantitatively and qualitatively observe and confirm the performance of SCA on shifted two-dimensional test functions.

Finally, the cross-section of an aircraft’s wing is optimized by SCA as a real challenging case study to verify and demonstrate the performance of this algorithm in practice.

The results of test functions and performance metrics prove that the proposed algorithm is able to explore different regions of a search space, avoid local optima, converge towards the global optimum, and exploit promising regions of a search space during optimization effectively.

The SCA algorithm obtains a smooth shape for the airfoil with a very low drag, which demonstrates that this algorithm can highly be effective in solving real problems with constrained and unknown search spaces (Mirjalili 2017).

The SCA algorithm theoretically is able to determine the global optimum of optimization problems due to the following reasons:

SCA creates and improves a set of random solutions for a given problem, so it intrinsically benefits from high exploration and local optima avoidance compared to other single-solution-based algorithms.

Different regions of the search space are explored when the sine and cosine functions return a value greater than 1 or less than -1.

Promising regions of the search space is exploited when sine and cosine return value between -1 and 1.

The SCA algorithm smoothly transits from exploration to exploitation using adaptive range change in the sine and cosine functions.

The best approximation of the global optimum is stored in a variable as the destination point and never get lost during optimization.

Since the solutions always update their positions around the best solution obtained so far, there is a tendency towards the best regions of the search spaces during optimization

Since the proposed algorithm considers optimization problem as black boxes, it is readily incorporable to problems in different fields subject to proper formulation of the problem (Mirjalili 2017).

4.8 Multi-Verse Optimizer

4.8.1 Inspiration

The big bang theory (Khoury J 2002) discusses that our universe starts with a massive explosion. According to this theory, the big bang is the origin of everything in this world, and there was nothing before that. The multi-verse theory is another recent and well-known theory among physicists (M 2004). It is believed in this theory that there is more than one big bang and each big bang causes the birth of a universe. The term multi-verse stands opposite of universe, which refers to the existence of other universes in addition to the universe that we all are living in (M 2004).

Multiple universes interact and might even collide with each other in the multiverse theory.

The multi-verse theory also suggests that there might be different physical laws in each of the universes.

We chose three main concepts of the multi-verse theory as the inspiration for the MVO algorithm: white holes, black holes, and wormholes. A white hole has never seen in our universe, but physicists think that the big bang can be considered as a white hole and maybe the main component for the birth of a universe (DM 1974). It is also argued in the cyclic model of multi-verse theory (Steinhardt PJ 2002) that big bangs white holes are created where the collisions between parallel universes occur. Black holes, which have been observed frequently, behave completely in contrast to the white wholes. They attract everything including light beams with their extremely high gravitational force (PC 1978). Wormholes are those holes that connect different parts of a universe together. The wormholes in the multi-verse theory act as time/space travel tunnels where objects are able to travel instantly between any corners of a universe (or even from one universe to another) (Morris MS 1988).

Every universe has an inflation rate that causes its expansion through space (AH 2007). The inflation speed of a universe is very important in terms of forming stars, planets, asteroids, black holes, white holes, wormholes, physical laws, and suitability for life. It is argued in one of the cyclic multi-verse models (Steinhardt PJ 2005) that multiple universes interact via white, black, and wormholes to reach a stable situation. This is the exact inspiration of the MVO algorithm, which is conceptually and mathematically modeled in the following subsection.

4.9 Moth-Flame Optimization Algorithm

In the anticipated MFO algorithm, we assume that the candidate solutions are the moths and their positions in space are variables of the problem. Consequently, the moths can fly in a 1-Dimensional, 2-Dimensional, 3-Dimensional, or hyper-dimensional area by altering their positions. As the MFO algorithm is a population-based procedure.

One point to be observed here is that both the moths and the flames are considered as solutions and both are updated and treated differently in every iteration.

The real search agents are the moths that fly in the search space while the finest spot of moths attained thus far is represented by the flames. Put it in another way, the flames are regarded as pins or flags which are released by the moths while looking through the search space. Consequently, every moth explores nearby a flame (flag) and revises it whenever it finds a superior solution. A moth will not miss its best solution by applying this procedure (YASIR ALI SHAH1 and NAWAZ3 2018).

Chapter 5

Experimental Results

5.1 Planning of a Three Units Power System

The three-unit system as a case study problem selected from (Saadat 2008) was utilized to explore the performance of CSA and DE as optimization methods to identify the best set of power generation of the unit power system. The adopted system is expected to produce a demand power of approximately 150 megawatts (MW). Table 5.1 displays the cost coefficients of the fuel of the three units system under investigation, or identified as P_1, P_2 and P_3 generators, and the coefficient matrix of the power loss (ζ) for the three units system are given next.

Table 5.1: Cost fuel coefficient of the three units system

P_i	α_i (\$/MW^2)	β_i (\$/MW^2)	γ_i (\$/MW^2)	P_{min} (MW)	P_{max} (MW)
P_1	0.0080	7.00	200	10	85
P_2	0.0090	6.30	180	10	80
P_3	0.0070	6.80	140	10	70

Table 5.2: Coefficient matrix of the power loss (ζ) for the three units

$$\zeta = \begin{bmatrix} 0.000218 & 0.000093 & 0.000028 \\ 0.000093 & 0.000228 & 0.000017 \\ 0.000028 & 0.000017 & 0.000179 \end{bmatrix}$$

Table 5.1 shows the computing power of the three units system based nine metaheuristic search algorithms with a demand power of 150 MW, where table 5.2 shows losses of transmission lines have been taken into account in these calculations.

The cost results are shown in Table 5.1 support that the load estimation methods - based CSA, and achieved better cost results than the algebraic method on the same three units power system, demonstrating their sensible capacities. The performance of proposed search algorithms for the three units power system is shown for up to 500 iterations in Figure 5.1. This convergence curve represents the fitness function created by nine meta-heuristic search algorithms for the power load estimation for three units' system.

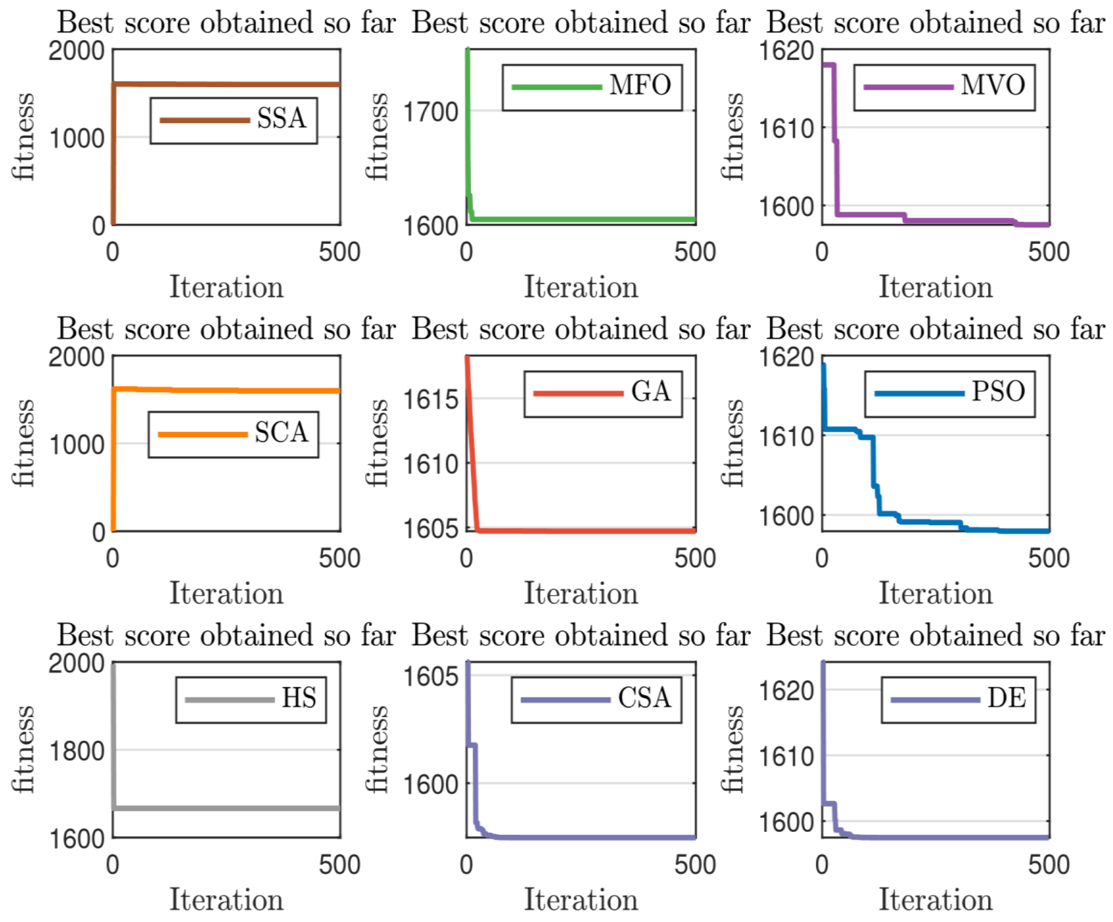

Figure 5.1: Three Units System: Convergence of evolutionary process of several metaheuristic search algorithms

Table 5.3: Cost Coefficient of Six Units System

	SSA	MFO	MVO	SCA	GA	PSO	HS	CSA	DE
P_1	35.52	23.9847	32.6064	27.2953	57.3072	25.5111	53.4052	32.8112	32.8101
P_2	59.71	80	65.0719	67.4894	72.084	61.0358	56.4331	64.5944	64.595
P_3	57.07	48.5765	54.6698	57.5698	23.5098	65.7379	43.1074	54.9365	54.9369
PP_i	152.29	152.56	152.35	152.35	152.90	152.28	152.95	152.34	152.34
P_D	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00	150.00
Cost ($/hr)	1597.83	1600.93	1597.53	1598.12	1612.05	1599.11	1656.28	**1597.48**	**1597.48**

5.2 Planning of a Six Units Power System

A set of experiments on six units' system consisting of six thermal power plant units was performed to illustrate the effectiveness of several meta-heuristics search algorithms in estimating the generation unit power. The prime goal is to find an estimate for the power load for each ith unit system, P_i so that the cost is reduced.

Table 5.4: Cost Coefficient of Six Units System

P_i	α_i ($/MW^2)	β_i ($/MW^2)	γ_i ($/MW^2)	P_{min} (MW)	P_{max} (MW)
P_1	0.0070	7.0	240.0	100	500
P_2	0.0095	10	200.0	50	200
P_3	0.009	8.5	220.0	80	300
P_4	0.009	11	200.0	50	150
P_5	0.008	10.5	220.0	50	200
P_6	0.0075	12	190.0	50	120

of the thermal units cost coefficient given in Table 5.3, and Table 5.4 show the coefficient matrix (ζ) is provided next.

Nine meta-heuristic search algorithms were executed to allocate the best performance and compute the estimated power load by each algorithm. The performance of proposed search algorithms for the three units power system is shown for up to 500 iterations in Figure 5.2. This convergence curve represents the fitness function created by nine meta-heuristic search algorithms for the power load estimation for six units system.

Table 5.5: Coefficient matrix of the power loss (ζ) for the six units

$$\zeta = \begin{bmatrix} 0.0170 & 0.0120 & 0.0070 & -0.0010 & -0.0050 & -0.0020 \\ 0.0120 & 0.0140 & 0.0090 & 0.0010 & -0.0060 & -0.0010 \\ 0.0070 & 0.0090 & 0.0310 & 0 & -0.0100 & -0.0060 \\ 0.0010 & 0.0010 & 0 & 0.0240 & -0.0060 & -0.0080 \\ -0.0050 & -0.0060 & -0.0100 & -0.0060 & 0.1290 & -0.0020 \\ -0.0020 & -0.0010 & -0.0060 & -0.0080 & -0.0020 & 0.1500 \end{bmatrix} \times 10^{-3}$$

Table 5.6: Optimal generations power of various algorithms

Power	SSA	MFO	MVO	SCA	GA	PSO	HS	CSA	DE
P_1	443.5555	500	452.8114	417.6776	288.9704	471.9346	409.6441	446.9736	447.5787
P_2	173.5464	200	182.2327	200	315.0599	187.8771	193.7739	173.319	173.0238
P_3	269.286	236.0095	263.0214	300	116.3195	272.913	285.2534	263.7248	263.9873
P_4	131.7837	150	135.484	150	160.0623	140.4767	147.215	138.9444	139.1728
P_5	182.4648	128.8926	152.5703	134.1542	259.6067	100.0305	124.6418	165.6265	165.0263
P_6	75.15023	60.3902	89.25042	73.43134	137.9263	102.2144	114.7046	86.8287	86.62046
PP_i	1275.787	1275.292	1275.37	1275.263	1277.945	1275.446	1275.233	1275.417	1275.409
P_D	1263	1263	1263	1263	1263	1263	1263	1263	1263
Cost ($/hr)	15447.54	15498.2	15445.69	15488.04	16135.5	15494.94	15490.75	**15442.66**	**15442.66**

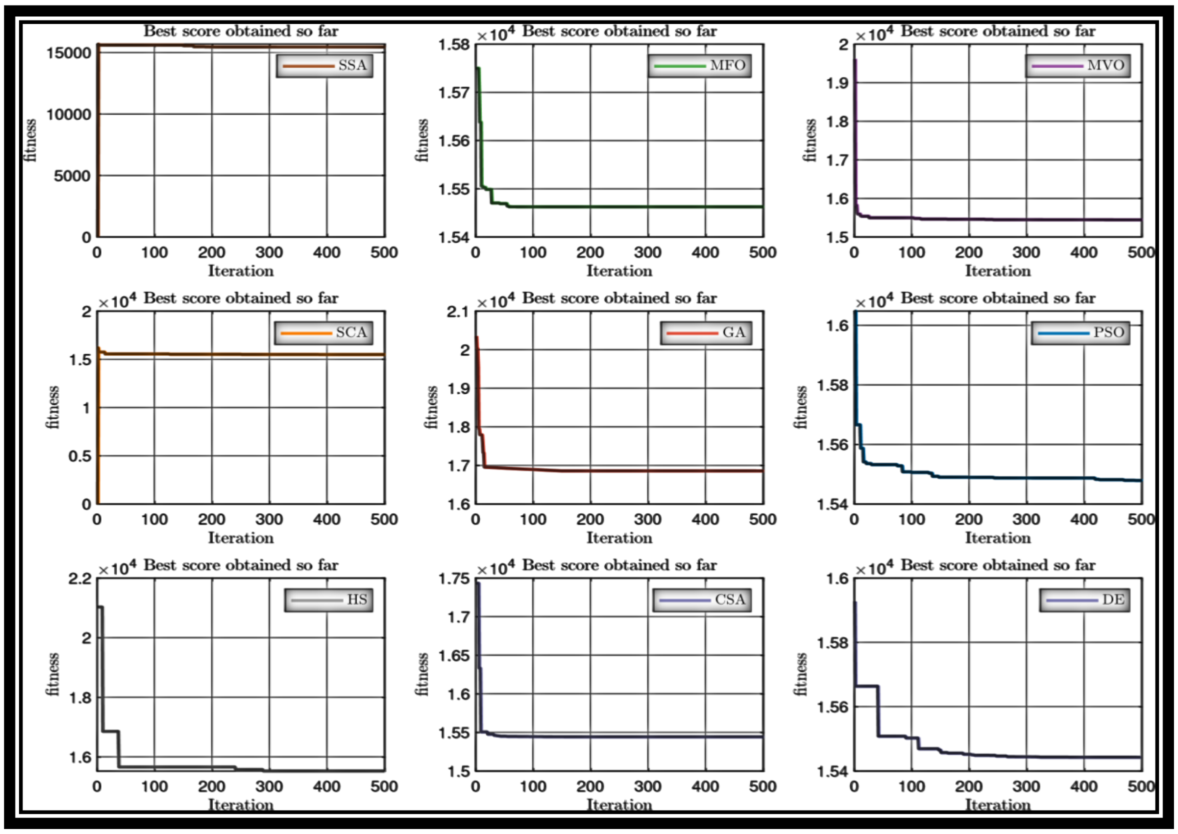

Figure 5.2: Six Units System: Convergence of evolutionary process of several metaheuristic search algorithms

5.3 Planning for the IEEE 30 Bus System

To further illustrate the efficacy of both CSA and DA in solving the ELD problem, both are practiced to a standard IEEE 30 bus consisting of a system of six units thermal power plant. The goal is to locate the best generated power of the *i*th generator, P_i, for the IEEE 30 Bus with six generator test system shown in Figure 5.7.

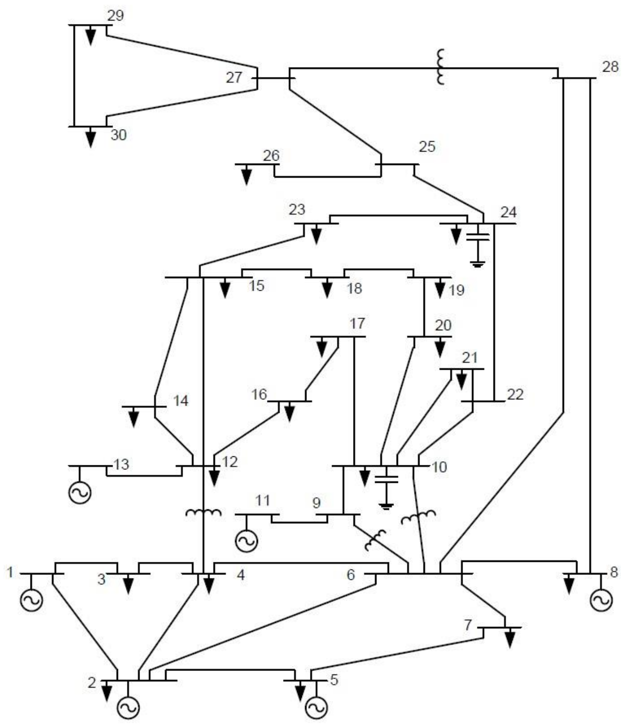

Figure 5.3: IEEE 30 Bus consisting of six generators test system

The thermal units characteristics of the IEEE 30 bus system are shown in Table 5.3 and the coefficient matrix (ζ) indicating the losses is introduced below.

Table 5.7: Cost Coefficient of IEEE 30 Bus System

P_i	α_i (\$/MW^2)	β_i (\$/MW^2)	γ_i (\$/MW^2)	P_{min} (MW)	P_{max} (MW)
P_1	15.240 $\times 10^{-2}$	38.53973 $\times 10^2$	756.79886	10	125
P_2	10.587 $\times 10^{-2}$	46.15916 $\times 10^2$	451.32513	10	150
P_3	2.803 $\times 10^{-2}$	40.39655 $\times 10^2$	1049.9977	35	225
P_4	03.546 $\times 10^{-2}$	38.30553 $\times 10^2$	1243.5311	35	210
P_5	2.111 $\times 10^{-2}$	36.32782 $\times 10^2$	1658.5596	130	325
P_6	1.799 $\times 10^{-2}$	38.27041	1356.6592	125	315

Table 5.8: Coefficient matrix of the power loss (ζ) for IEEE 30 Bus System

$$\zeta = \begin{bmatrix} 0.1400 & 0.0170 & 0.0150 & 0.0190 & 0.0260 & 0.0220 \\ 0.0170 & 0.0600 & 0.0130 & 0.0160 & 0.0150 & 0.0200 \\ 0.0150 & 0.0130 & 0.0650 & 0.0170 & 0.0240 & 0.0190 \\ 0.0190 & 0.0160 & 0.0170 & 0.0710 & 0.0300 & 0.0250 \\ 0.0260 & 0.0150 & 0.0240 & 0.0300 & 0.0690 & 0.0320 \\ 0.0220 & 0.0200 & 0.0190 & 0.0250 & 0.0320 & 0.0850 \end{bmatrix} \times 10^{-3}$$

It is evident from Table 5.3 that the CSA and DE-based methods reported the best performance regarding the best load results. The computed values for each power unit P_i (i = 1,2,...,6) for the standard IEEE 30 Bus shown in Table 5.3. The performance of proposed meta-heuristic search algorithms for the IEEE30 Bus system is shown for up to 500 iterations in Figure 5.4. This convergence curve represents the fitness function created by nine meta-heuristic search algorithms for the power load estimation for IEEE30 Bus system.

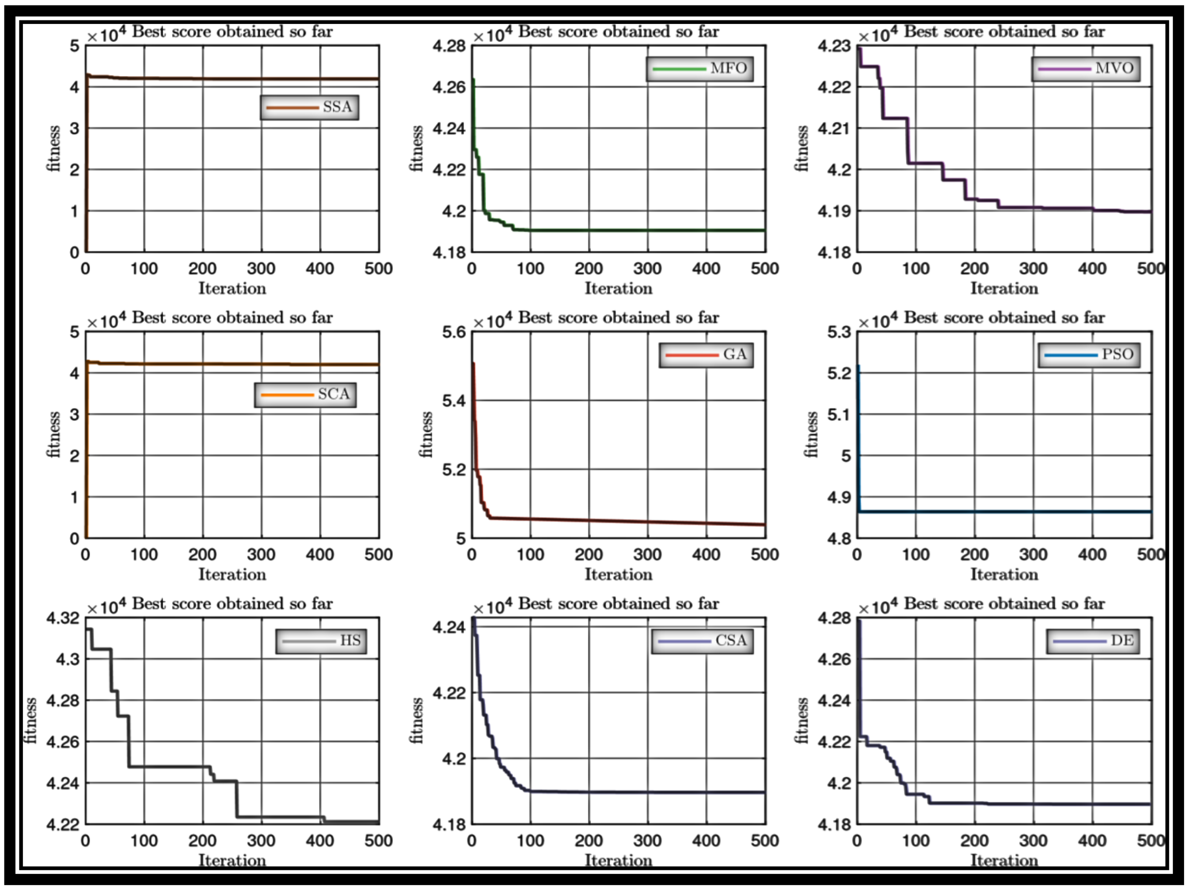

Figure 5.4: IEEE 30 Bus: Convergence of evolutionary process of several metaheuristic search algorithms

Table 5.9: Cost Coefficient of IEEE 30 Bus system

Power	SSA	MFO	MVO	SCA	GA	PSO	HS	CSA	DE
P_1	31.3561	27.71748	33.05922	23.49077	120.4477	125	21.56365	32.59279	32.60999
P_2	13.29429	21.9596	13.88867	14.34295	103.4747	125	105.1503	14.49964	14.36192
P_3	144.0152	130.5341	139.9768	135.2147	115.2445	125	204.1934	141.6495	141.6919
P_4	135.9548	125.1346	134.5273	157.3899	124.2421	125	100.7099	136.0354	135.7933
P_5	263.4618	255.0508	262.7322	262.9998	124.5823	125	193.4693	257.5987	257.7318
P_6	237.2559	265.3226	241.2151	231.2804	121.2529	125	196.193	242.9515	243.1477
PP_i	825.338	825.7192	825.3993	824.7184	709.2442	750	821.2795	825.3275	825.3366
P_D	800	800	800	800	800	800	800	800	800
Cost ($/hr)	41898.68	41925.3	41897.52	41962.89	50023	48639.58	43143.66	41896.63	41896.63

The results are compared in terms of the operating cost of generators and power generation. Wide simulation results are observed to minimum operation cost, minimum standard deviation among best, mean, and worst solution showing that both CSA and DE provided good ex-portability, fast convergence with iteration leads to robustness and good solution quality.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we provided an innovative model to manage a smart electric power grid to increase the quality of top service and reduce the cost of operation. This is a complex adaptive system design problem for distributed power generation. This work focused on solving optimization problems in the smart grid by using Metaheuristic search algorithms. This research explored the Economic dispatch (ED) problem aiming to distribute the load demand between all of the various generation units in an electrical system such that the total cost of generation is very minimum. To solve the ED problem, we used nine search algorithms and compared their results. The efficiency and effectiveness of the nine techniques are bench-marked for different test cases consisting of IEEE 30 bus, three, six for generating units with high nonlinearity. The results are compared in terms of the operating cost of generators and power generation. Wide simulation results are observed to minimum operation cost, minimum standard deviation among best, mean, and worst solution showing that both CSA and DE provided good ex-portability, fast convergence with iteration leads to robustness and good solution quality.

6.2 Future Work

In the future, we hope to continue to solving the power generation problem area like unit commitment problems by apply on Meta-heuristics algorithm and explores the best minimums fuel cost.

Bibliography

A. Abraham, C. Grosan, V. R. (2008). Swarm intelligence in data mining. *Studies in Computational Intelligence.*.

A. J. Wood, B. F. W. (2006). New york: John wiley sons, inc., second ed. *Power Generation Operation and Control*.

A. Walker, J. Hallam, D. W. (1993). Ieee service center. *the construction of aselforganized cognitive map and its use in robot navigationwithin acomplex.*, 1451–1456.

Abbassi, R., A. A. H. A. A. M. S. (2019). Energy conversion and management. *An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. 179*, 362–372.

AH, G. (2007). Eternal inflation and its implications. *J Phys A Math Theory.* (40).

Alba, E. (2005). Wiley-interscience. *Parallel Metaheuristics: A New Class of Algorithms.*.

Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. *Evolutionary Programming VII.*.

Aristidis, V. (2006). An ant colony optimization (ACO) algorithm solution to economic load dispatch (eld) problem. In *Proceedings of the 10th WSEAS International Conference on Systems*, ICS'06, Stevens Point, Wisconsin, USA, pp. 153– 160. World Scientific and Engineering Academy and Society (WSEAS).

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures 169*, 1–12.

Ba¨ck, T. and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation 1*(1), 1–24.

Bergen, A. R. (1986). *Power systems analysis*. Prentice-Hall series in electrical and computer engineering. Englewood Cliffs (N.J.): Prentice Hall.

Bhattacharya, A. and P. K. Chattopadhyay (2010, May). Solving complex economic load dispatch problems using biogeography-based optimization. *Expert Syst. Appl. 37*(5), 3605–3615.

Bhattacharya, A. and P. K. Chattopadhyay (2011, March). Solving economic emission load dispatch problems using hybrid differential evolution. *Appl. Soft Comput. 11*(2), 2526–2537.

C, B. and R. A (2003). Acm comput. *Metaheuristics in combinatorial optimization: Overview and conceptural comparison. 35*, 268–308.

C. Koulamas, S.R. Antony, R. J. (1994). Omega. *A survey of simulated annealing applications to operations research problems. 22*, 41–56.

Cerny, V. (1985). Thermo dynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Optimization Theory andApplications* (45), 41–51.

Chakraborty, P., G. G. Roy, B. K. Panigrahi, R. C. Bansal, and A. Mohapatra (2012). Dynamic economic dispatch using harmony search algorithm with modified differential mutation operator. *Electrical Engineering 94*(4), 197–205.

Chen, P.-H. and H.-C. Chang (1995). Large-scale economic dispatch by genetic algorithm. *IEEE Transactions on Power Systems 10*, 1919–1926.

Das, S. and P. N. Suganthan (2011, Feb). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation 15*(1), 4–31.

D.E. Goldberg, K. D. (1991). Foundations of genetic algorithms. *A comparative analysis of selection schemes used in genetic algorithms.* (4), 69–93.

Dewangan, S. K., A. Jain, and A. P. Huddar (2010, April). Comparison of particle swarm optimization with lambda iteration method to solve the economic load dispatch problem. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 4*(4), 1900–1907.

DM, E. (1974). Phys rev. *Death of white holes in the early Universe.* (33).

Dorigo, M. (1992). Politecnico di milano. *Optimization, Learning and Natural Algorithms.*.

(Ed.), C. T. (2008). In-tech education and publishing. *Simulated Annealing, first ed.*.

Engelbrecht, A. (2006). Fundamentals of computational swarm intelligence. *Wiley.*.

Fleischer, M. (1995). Arlington, va, usa. *Simulated annealing: past, present and future, in: Proceedings of the 27th Conference on Winter Simulation.*, 155–161.

F.T.S. Chan, M. T. (2007). Swarm intelligence, focus on ant and particle swarm optimization. *I-Tech Education and Publishing.*.

Gaing, Z. L. (2003). Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Transactions on Power Systems 18*, 1187–1195.

Geem ZW, K. J. and L. GV (2001). Simulation. *A new heuristic optimization algorithm: Harmony search. 76*, 60–68.

Glover, F. (2008). Computers and operations research. *Future paths for integer programming and links to artificial intelligence.* (13), 533–549.

Goldberg, D. (1989). Studies in computational intelligence. *Genetic Algorithms in Search, Optimization, and Machine learning.*.

H.G. Beyer, H. S. (2002). Natural computing. *Evolution strategies – a comprehensive introduction. 1*, 3–25.

Holland, J. (1975a). University of michigan press. *Adaptation in Natural and Artificial Systems.*.

Holland, J. H. (1975b). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, University of Michegan Press (2nd ed.: MIT Press, 1992).

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press.

J. Kennedy, R. E. (1995). Ieee international conference on neural networks. *Particle swarm optimization.*, 1942–1948.

J. Kennedy, R. M. (2002). Proceedings of the world on congress on computationalintelligence. *Population structure and particle swarm performance. 2*.

J.D. Farmer, N.H. Packard, A. P. (1986). Physica d 2. *The immune system, adaptation, and machine learning.*, 187–204.

Khoury J, Ovrut BA, S. N. S. P. T. N. (2002). *From big crunch to big bang*. Phys Rev D.

Kirchmayer, L. (1979). New delhi: Wiley eastern limited, first ed. *Economic Operation of Power Systems*.

Koza, J. (1992). The mit press. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems).*.

Koza, J., F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap (1997). Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 109–128.

KS, L. and G. ZW (2005). Comput. methods appl. mech. engrg. *A new metaheuristic algorithm for continuous engineering optimization: harmony search theory and practice. 194*, 3902–3933.

Kuo, C. C. (2008). A novel coding scheme for practical economic dispatch by modified particle swarm approach. *IEEE Transactions on Power Systems 23*, 1825–1835.

K.V. Price, R.M. Storn, J. L. (2005). Lampinen, differential evolution a practical approach to global optimization. *Natural Computing Series.* (11), 341–359.

L.J. Fogel, A.J. Owens, M. W. (1966). John wiley. *Artificial Intelligence through Simulated Evolution.*.

M, D. and B. C (2005). Comput. sci. *Ant colony optimization theory: A survey. Theor. 344*, 243–278.

M, O. and Mahdavi (2008). Applied math.computation. *Global-best harmony search. 198*, 643–656.

M, T. (2004). Science and ultimate reality: Quantum theory, cosmology, and complexity. cambridge university press. *Parallel universes. In: Barrow JD, Davies PCW, Harper CL Jr (eds).*, 459–491.

Mirjalili, S., G. A. H. M. S. Z. S. S. F. H. . M. S. M. (2017). Advances in engineering software. *Salp swarm algorithm: A bio-inspired optimizer for engineering design problems.*.

Mirjalili, S. Knowledge-based systems. *SCA: A Sine Cosine Algorithm for Solving Optimization Problems*.

Morris MS, T. K. (1988). a tool for teaching general relativity. *Wormholes in space time and their use for interstellar travel.* (56), 395–412.

N. Hansen, A. Ostermeier, A. G. (1995). Proceedings of the 6th international conference on genetic algorithms. *On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation.*, 57–64.

N. Metropolis, A. Rosenbluth, M. R. A. E. T. (1953). Equation of state calculationsby fast computing machines. *Chemical Physics* (21), 1087–1090.

N.E. Collins, R.W. Eglese, B. G. (1988). American journal of mathematical and management sciences. *Simulated annealing – an annotated bibliography. 8*, 209–307.

Nguyen, T. T. and D. N. Vo (2015, December). The application of one rank cuckoo search algorithm for solving economic load dispatch problems. *Appl. Soft Comput. 37*(C), 763–773.

P. R. Lorenzo, J. Nalepa, M. K. L. S. R. J. R. P. (2017). Particle swarm optimizationfor hyper-parameter selection in deep neural networks. *Proceedings of the Genetic and Evolutionary Computation Conference.*, 481–488.

Passino, K. (2002). Ieee control systems magazine. *Biomimicry of bacterial foraging for distributed optimization and control* (22), 52–67.

PC, D. (1978). Thermodynamics of black holes. *Rep Prog.* (41).

Poli, R., J. Kennedy, and T. Blackwell (2007). Particle swarm optimization. *Swarm Intelligence 1*(1), 33–57.

P.V. Laarhoven, E. A. (1987). Reidel publishing company. *Simulated Annealing: Theory and Applications, first ed., D.*.

Rahmani, R., M. F. Othman, R. Yusof, and M. Khalid (2012, December). Solving economic dispatch problem using particle swarm optimization by an evolutionary technique for initializing particles. *Journal of Theoretical and Applied Information Technology 46*(2), 526–536.

R.C. Eberhart, J. K. (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science.*.

Rechenberg, I. (1973). Frommann-holzboog. *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution.*.

R.M. Storn, K. P. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Global Optimization.* (11), 341–359.

S. Das, A. Abraham, A. K. (2008). Swarm intelligence algorithms in bioinformatics. *Computational Intelligence in Bioinformatics, Studies in Computational Intelligence. 94*, 113–147.

S. Kirkpatrick, C. Gelatt, M. V. (1983). Science 220. *Optimization by simulated annealing.*, 671–680.

Saadat, H. (2008). *Power System Analysis*. McGraw-Hill Companies.

Sahoo, S., K. M. Dash, R. Prusty, and A. Barisal (2015). Comparative analysis of optimal load dispatch through evolutionary algorithms. *Ain Shams Engineering Journal 6*(1), 107 – 120.

Sen, D. and P. Acharjee (2016, Oct). Hybridization of cuckoo search algorithm and chemical reaction optimization for economic load dispatch problem. In *Proceedings of the 2016 International Conference and Exposition on Electrical and Power Engineering (EPE)*, pp. 798–804.

Simon, D. (2008). Ieee transactions on evolutionary computation. *Bio geographybased optimization* (12), 702–713.

Steinhardt PJ, T. N. (2002). A cyclic model of the universe. *Science.* (296), 1436–1439.

Steinhardt PJ, T. N. (2005). The cyclic model simplified. *New Astron Rev.* (49), 43–57.

T. Blickle, L. T. (1995). Evolutionary computation. *A comparison of selection schemes used in genetic algorithm.* (4), 311–347.

Wood, A. J. and B. F. Wollenberg (2010). Power Generation Operation and Control (2nd Edition).

Yang, X.-S. (2008). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

YASIR ALI SHAH1, 2, H. A. H. F. A. . M. F. K. . M. M. and T. NAWAZ3 (2018, September). Ieee. *CAMONET: Moth-Flame Optimization (MFO) Based Clustering Algorithm for VANETs.*.

ye, F. (2017). Particle swarm optimization-based automatic parameter selection for deep neural networks. *applications in large-scale and high-dimensional data. 12*, 12.

ZW, G. (2006). Engineering optimization. *Optimal cost design of water distribution networks using harmony search. 38*, 259–280.

ZW, G. (2007). Lecture notes in computer science. *Optimal scheduling of multiple dam system using harmony search algorithm. 4507*, 316–323.

ZW, G. (2008). Harmony search for solving sudoku. *Knowledge-Based Intelligent Information and Engineering Systems. 4692*, 371–378.