University of Southampton

Faculty of physical Science and Engineering

School of Electronics and Computer Science

Kriging Assisted Evolutionary Computation in Power System Optimization

by

Zhida DENG

Thesis for the degree of Doctor of Philosophy

June 2020

University of Southampton

ABSTRACT

Faculty of physical Science and Engineering

School of Electronic and Computer Science

Thesis for the degree of Doctor of Philosophy

Kriging Assisted Evolutionary Computation in Power System Optimization

by

Zhida DENG

Optimal Power Flow (OPF) is an important decision support tool for system operator to achieve reliable and economic processes in power system operation, control, market and planning. It provides system operators with optimal decisions and necessary control changes to achieve desired operational objectives, such as economically dispatching active power generations, minimizing active power losses, while satisfying load demand and specified constraints. However, OPF has proven to be the very difficult optimization problem due to its nonlinearity, nonconvexity and high constraints, while the current optimization algorithms cannot satisfy both computational speed and accurate solution simultaneously. Although the traditional method is widely applied to solve the practical OPF problem due to its computation efficiency, such technique is a compromise between computation efficiency and solution quality, producing local optimum and resulting in a large amount of unnecessary operatorial cost.

Alternative to the traditional method, the evolutionary computation method has attracted a lot of attentions in academic research because of its capability of finding the global optimum, however, the impractical computation time has limited its application especially in real-time applications. In order to improve the computation performance of evolutionary computation while saving the operational cost, the Kriging surrogate method has been found as a potential technique and not been applied to power system optimization in the past.

To achieve the aim, the work focuses on three topics related to the computational efficiency and solution quality: 1) the effective strategy used to alleviate the computation burden associated to Kriging, 2) the balance between exploitation and exploration during the optimization process; 3) the effective and efficient strategy used to manage the Kriging in evolutionary computation. The effectiveness and the performance of proposed algorithms are verified on the complex multimodal optimization functions and benchmark power systems with results compared with the algorithms reported in literature.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AC Alternative Current

ACE Area Control Error

AGC Automatic Generation Control

ARC Automatic Remedial Control

AVC Automatic Voltage Control

DAUC Day-Ahead Unit Commitment

DC Direct Current

DMS Distribution Management System

ED Economic Dispatch

EGO Efficient Global Optimization

EI Expected Improvement

ELS Elitist Learning Strategy

EMS Energy Management System

FA Factor Analysis

FACTS flexible ac transmission system

FEs Function Evaluations

GA Genetic Algorithm

GAMS General Algebraic Modelling System

GLS Generalized Least Square

GMEI Generalized Maximum Expected Improvement

GP Gaussian Process

HVAC High-Voltage Alternative Current

HVDC High-Voltage Direct Current

IPM Interior Point Method

KAGA Kriging Assisted Genetic Algorithm

KAPSO Kriging Assisted Particle Swarm Optimization

KELS Kriging assisted Elitist Learning Strategy

KKT Karush–Kuhn–Tucker

KOEDLS Kriging assisted Orthogonal Experiment Design Learning Strategy

LP Linear Programming

MINLP Mixed-Integer Nonlinear Programming

MSE Mean Squared Error

NLP Nonlinear Programming

OA Orthogonal Array

OARPD Optimal Active and Reactive Power Dispatch

OED Orthogonal Experiment Design

OLS Ordinary Least Square

OPF Optimal Power Flow

ORPD Optima Reactive Power Dispatch

OWPP Offshore Wind Power Plant

p.u. per unit

PKAGA Pre-selection Kriging Assisted Genetic algorithm

PKAPSO Pre-selection Kriging Assisted Particle Swarm Optimization

PR Polynomial Regression

PSO Particle Swarm Optimization

QP Quadratic Programming

RBF Radial Basis Functions

RPE Reactive Power Enforcement

RT Real-Time

RTED Real-Time Economic Dispatch

RTOPF Real-Time Optimal Power Flow

RTU Remote Terminal Units

SAEC Surrogate Assisted Evolution Computation

SCADA Supervisory Control and Data Acquisition

SCED Security Constrained Economic Dispatch

SCOPF Security-Constrained Optimal Power Flow

SDP Semidefinite Programming

SLP Sequential Linear Programming

SM Simplex Method

SOCP Second Order Cone Programming

SQP Sequential Quadratic Programming

SR Success Rate

SSR Sum of Squared Residuals

Std. Standard Deviation

TSCOPF Transient Stability-Constrained Optimal Power Flow

UC Unit Commitment

WMEI Weighted Maximum Expected Improvement

List of Symbols

$\lceil \cdot \rceil$: Celling operator

$\lfloor \cdot \rfloor$: Floor operator

$\hat{\sigma}^2$: Optimal variance of Gaussian Process

$\| \cdot \|$: Euclidean Norm

$\hat{\mu}$: Optimal mean value of Gaussian Process

c_1: personal (or cognitive) learning coefficient

c_2: global (or social) learning coefficient, respectively

$\boldsymbol{G}_{best}$: Global best

$\boldsymbol{P}_{best}$: Personal best

$\boldsymbol{Q}_{sh}$: Shunt compensator

∇: Derivative operator

exp: Exponential operator

log: Logarithm operator

PQ bus: Load bus

PV: Generator bus

Tr(·): Trace operator

Φ: Cumulative distribution function

$N(0, \sigma)$: normally distributed random number

T: Transpose operator

aw_KELS: Award used to trigger the use of KELS

aw_OA: Award used to trigger the use of real fitness factor analysis

aw_WEI: Award used to assign the weight to EI

nc: Number of candidate solutions

ns: Number of samples

p: Population/swarm size

$\boldsymbol{J}$: Jacobian matrix

$\boldsymbol{L}$: Line power flow

$\boldsymbol{P}$: Active power

$\boldsymbol{Q}$: Reactive power

$\boldsymbol{R}$: Correlation matrix containing pairwise distances between pairs of samples

$\boldsymbol{T}$: Tap ratio

$\boldsymbol{V}$: Voltage magnitude

$\boldsymbol{\mathscr{r}}$: Correlation matrix containing pairwise distances between samples and estimations

σ: Std. of Gaussian distribution

ω: Inertia weight

∂: Differential operator

ϕ: Probability density function

$\boldsymbol{\delta}$: Voltage angle

Research Thesis: Declaration of Authorship

Print name: Zhida DENG

Title of thesis: Kriging Assisted Evolutionary Computation in Power System Optimization

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:

   Z. Deng, M. D. Rotaru and J. K. Sykulski, "A study of evolutionary based optimal power flow techniques," 2016 51st International Universities Power Engineering Conference (UPEC), Coimbra, pp. 1-6.

   Z. Deng, M. D. Rotaru and J. K. Sykulski, "Harmonic Analysis of LV distribution networks with high PV penetration," 2017 International Conference on Modern Power Systems (MPS), Cluj-Napoca, pp. 1-6.

   Z. Deng, M. D. Rotaru and J. K. Sykulski, "Kriging Assisted Surrogate Evolutionary Computation to Solve Optimal Power Flow Problems," in IEEE Transactions on Power Systems, vol. 35, no. 2, pp. 831-839, March 2020.

Signature: _____ Date:_____

Acknowledgements

Thanks firstly to my family and my girlfriend for providing fully and selfless supports throughout the process of my Ph.D. study.

Thanks to my supervisors Professor Jan Sykulski and Dr Mihai Rotaru; without their encouragements and valuable suggestions, this would not happen.

Thanks to our group leader Professor George Chen for his kindly and generous help over the past years.

Thanks to my examiner Dr James Pilgrim for his helpful advice about my research in my 9 month and transfer reports.

Thanks to all my friend met during these years for their companying and colouring my life.

Chapter 1 Introduction

1.1 Electric Power System

Electrification is the one of most significant developments during the Secondary Industry Revolution and it is also the key element in modern societies. Electric power has changed the world in many different ways where from small portable devices to large manufacture machines are powered by electricity. Nowadays life without electricity is difficult to conceive and our daily life significantly relies on it. The system responsible for generating, transmitting and distributing electricity is called electric power system. The first electric power system can be traced back to 1880's, which was built by Thomas Edison and generating Direct Current (DC). However, the DC power system at that time would never be economical to scale up due to the limitation of transmission radius caused by voltage drop. With the early advantage of transformer, Nicola Tesla proposed Alternative Current (AC) power system, which can transmit electricity over long distance and won the "War of Currents". Eventually, the AC power system has become primary power system around the world. Recent decades, with the development of advanced power electronic technologies, DC has made a comeback and some High-Voltage DC (HVDC) transmission projects have been installed and planned for future network expansions [1], due to the low transmission losses (e.g. absence of reactive power), the high reliability and the flexible controllability (e.g. independent active and reactive power control) [2]. Although the DC transmission technology is currently seen as a strong competitor for High-Voltage AC (HVAC) transmission and it has bright future and advantages in certain applications, such as long distance transmission (e.g. DC is more economical than AC when power ratings greater than 1,000 MW and distance over 600 KM [3]), integrating large scale Distributed Energy Resources (DERs) and interconnecting regional asynchronous AC power systems [2], [3], it is difficult to assert when the power system would be fully dominated by DC (especially at low-voltage networks) because of the high investment costs and the lack of reliable commercial DC protection devices [4]. Under this concern, this work only considers the AC power system.

After 130 years development, the power system has experienced tremendous evolution; it can supply electricity to users 24/7 without interruption unless contingency occurs. In fact, this is the result of many coordinated efforts– such as sufficient generation for meeting the demand, robust infrastructure for delivering electricity, proper standard for regulating the system, effective strategy for operating and controlling the system, advanced instrument for monitoring the system, and advanced devices for protecting the system. These concerns are worth to be studied for improving power system efficiency and reliability, but it is impossible to study all of them in this work. Nevertheless, this work mainly focuses on power system operation, because it is the core of coordinating whole system and it is one of most important concerns for ensuring the system efficiency and reliability.

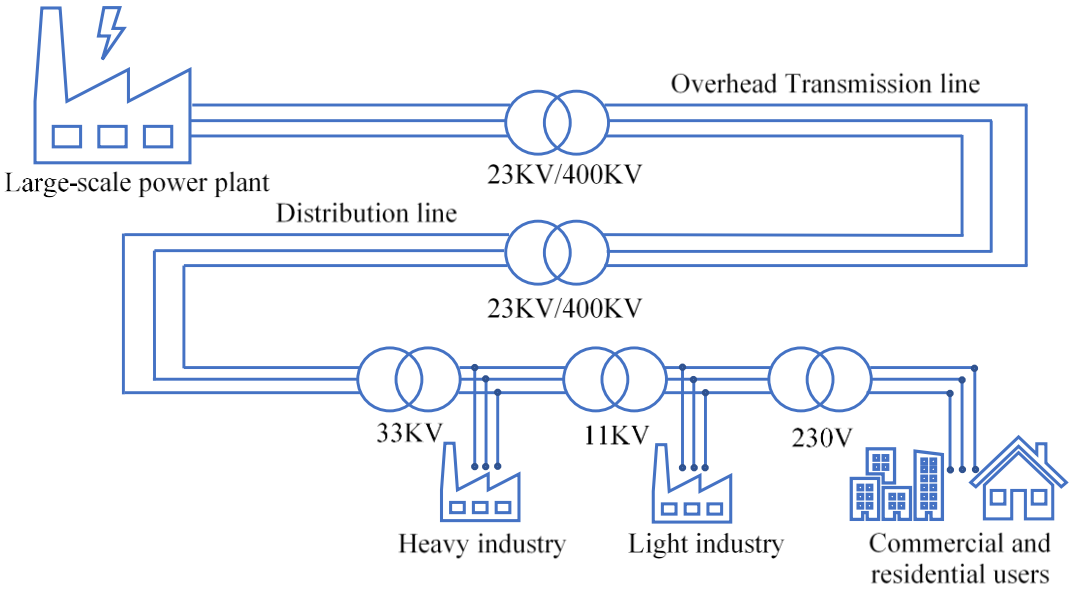


Fig. 1.1 A simple illustration of UK interconnected electric power system [5]

An electric power system consists of many components, which can be generally divided into following four major parts:

- Generation,
- Transmission
- Distribution
- Load

As shown in Fig. 1.1 [5], this kind of power system structure has unidirectional power flow, which the electricity is transmitted over a country or region from large-scale generation stations to loads through high-voltage transmission girds and low-voltage distribution networks. Electricity is a secondary energy that has to be converted from primary energy in generation power plant. The primary energy sources – such as fossil fuels, nuclear power, hydropower, renewable energies, drive the turbine and convert mechanical energy into electricity energy. Due to the economic and environmental concerns, the traditional large-scale power plants are usually constructed far away from the load centre. Therefore the high-voltage transmission grids are necessary for long distance transmissions and reducing transmission losses, according to the apparent power equation – transmitting same amount of power, the higher the voltage level the lower the current flow (i.e. power losses) tend to be. For safety purpose, the voltage must be stepped down, which allows the electricity distributed to various type of loads via low-voltage distribution networks. However, in recent years the DERs have rapidly increased and widely installed at the low-voltage distribution networks to supply the local demands, due to the low investment cost, the environment concerns and the sustainability [6]. This has changed the traditional power system structure to bidirectional power flow, which the electricity could also be transmitted from distribution networks or load sides to other

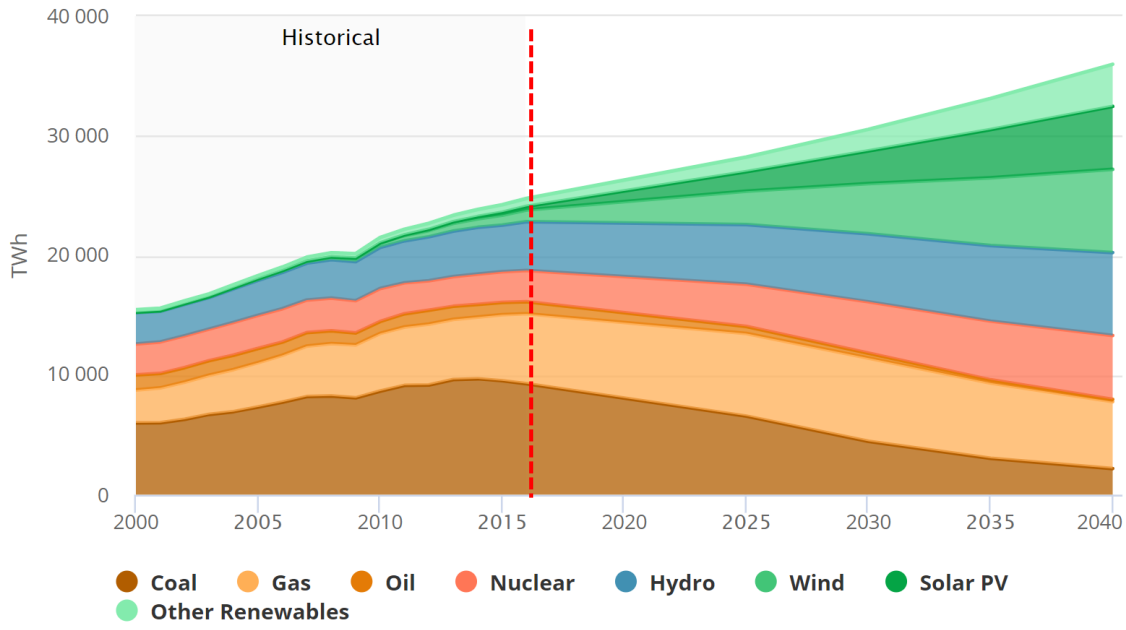


Fig. 1.2 International Energy Agency's World Energy Outlook 2017 [7]

region via high-voltage transmission girds. In addition to these four major parts, many other controllable components like transformer, circuit break, and compensator shunt exist in the power systems. These make the power system operation not trivial, as one may need to coordinate the movements of many components to meet the constantly changing demand and maintain the system efficiency and reliability.

Currently electricity in worldwide is mainly generated by using fossil fuels, nuclear power and hydropower. However, the depletion of fossil fuels, environment pollutions and climate changes have raised serious concerns on sustainability of power system. The use of renewable energy sources has been proposed as an efficient way to alleviate and address these issues. The future electric power systems will be subsequently tending to accommodate high penetration level of renewables. In 2016, the worldwide electricity consumption from renewable sources is around 8% of total consumption and it is expected that will be changing to 45% in 2040 [7], as shown in Fig. 1.2. However, the output of renewable generations highly depends on weather conditions, which is not controllable and manageable like traditional generations. Their intermittent nature consequently brings critical challenges to power system operation and control, such as difficult to meet the uncertain demand and maintain the system reliability at all time. In order to ensure the system reliability while maintaining the system efficiency, the power system operation and control must be robust and responding quickly to system changes. In other word, power system needs to be adjusted more frequently to deal with uncertainties from both generation and demand sides.

1.2 Power System Control Centre

The operational and control decisions are usually made in the control centre with the aid of various decision support tools in order to ensure system-wide reliability and efficiency. In the first half 20th century, these decisions were made by expert engineers and experienced operators using judgements, rules of thumb and specialized operation book [8]. This was a primitive and unreliable way for power systems operation. In 1960s, computer technology and optimization theory were introduced in control centre for improving efficiency of power system operation in real-time. Remote Terminal Units (RTUs) were developed and employed to collect real-time measurement system data to centre computer to calculate command signals for executing control requests. For example, Automatic Generation Control (AGC), which is a combination of Load Frequency Control (LFC) and Economic Dispatch (ED) [9]. This application of data acquisition, communication and supervisory control in control centre is called Supervisory Control and Data Acquisition (SCADA) system. The block diagram of power system AGC is illustrated in Fig. 1.3, where the LFC calculates Area Control Error (ACE) every few seconds to determinate regulation requirement and the ED (e.g. an optimization tool) provides optimal generation setpoints for participated generators to minimize total generation cost. AGC may also be called "Load-Following Control" that adjusts the generation to instantaneously satisfy the time-varying demand while maintaining the scheduled interchange flows and desired frequency from second to second, which has become the primary function in control centre. In 1970s, the capability of control centre was pushed to a new level to further improve the system reliability and efficiency with the introduction of Energy Management System (EMS), which is equipped with advanced system analysis software and optimization software in addition to

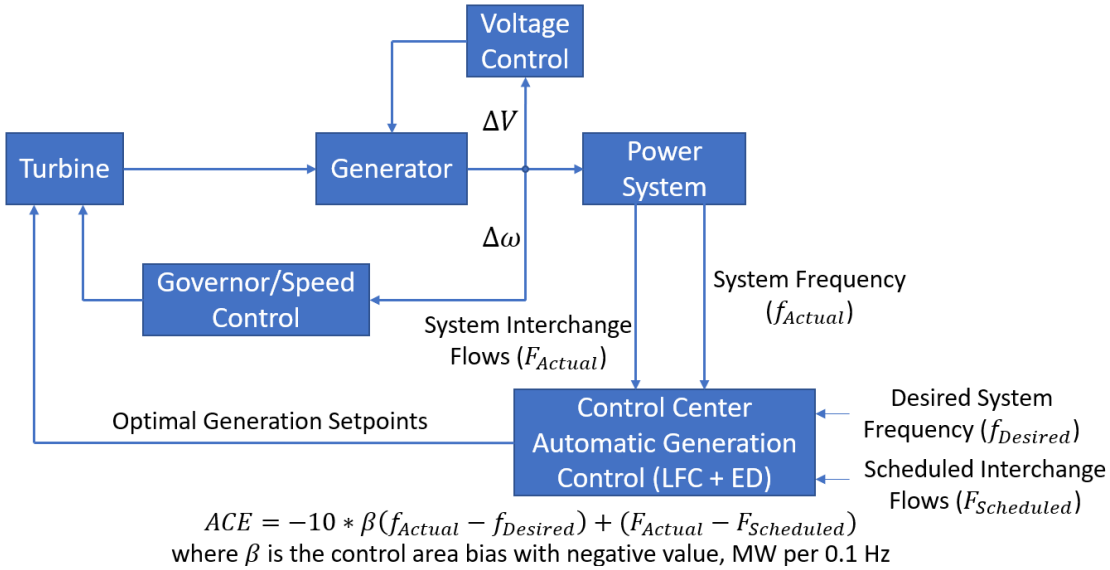


Fig. 1.3 Power System Automatic Generation Control

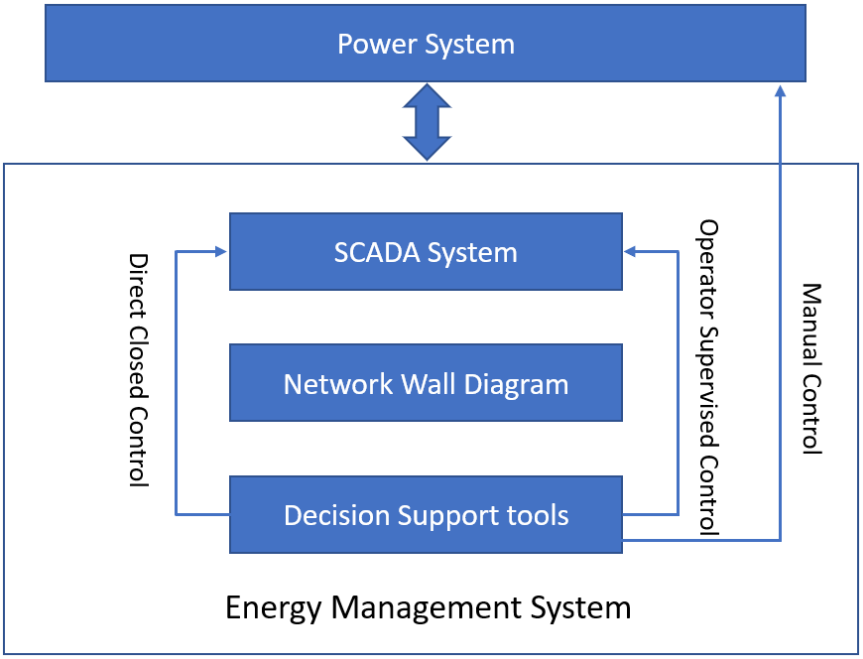


Fig. 1.4 Structure of Energy Management System

generation control software. The framework of modern EMS is shown in Fig. 1.4, which can be simply divided into three categories [10]:

- **System monitoring**: The SCADA system within the EMS continually gathers and updates measurement data of voltage, current, real and reactive power flows and circuit breakers status at power plants, substations and feeders. These data are made available to system operators to have instantaneous views of real-time operating status of power system via computer monitors and the network wall diagram (e.g. a large wall screen in control centre).
- **Decision support tools**: This portion of the EMS consists of many softwares, where the optimization package is the primary one for providing operators with control means to achieve operational objectives while meeting the system constraints, such as minimizing – generation cost, power losses, load shedding, and so on. Moreover, the reliability assessment package is another important tool of the EMS, which assesses static and transient stabilities of power system to identify the violations. The optimization and reliability assessment packages in fact are interconnected so that the optimization tool offers optimal solution while satisfying static and transient constraints and it is able to provide remedial actions for operators.
- **Control**: The issued control decisions are sent to participated devices via SCADA systems or other communication channel. The control actions can be classified into three types as follows:
  - *Direct-closed control:* These are automatic control actions, which are implemented by EMS without operation's participation, such as the Automatic Generation Control, Automatic Voltage Control (AVC) and designed Automatic Remedial Control (ARC).

- *Operator supervised control:* These controls are implemented by operators through EMS, for example the operators remotely increase or decrease generation outputs and open or close circuit breakers or compensator shunts.
- *Manual Control*: These control actions are implemented by the local staffs after receiving orders from system operators. For instance, the system operator calls a power plant to start -up or shut-down the generator and a substation to manually open or close the connector. This because the operator has no authority to directly control these devices.

More detailed functionality of EMS and the evolution of control centre may be found in [9]–[11]. Furthermore, similar solutions for distribution networks called Distribution Management System (DMS) have been developed to actively deal with the high penetration of distribution generations.

1.3 Power System Operation and Optimization

Power system operation in practice is an onerous task that the operators need various decision support tools to adjust system conditions and coordinate many controllable components' movements for ensuring the system operates in a reliable and efficient manner. The primary goal of power system operation is no doubt reliability, which refers to two attributes – adequacy and security. Adequacy means that the power system always has sufficient generation to meet the uncertain demands at all time; security means that the power system is operated within acceptable operational and physical constraints (e.g. voltage limits, thermal limits, stability, etc) and is able to withstand any contingency without violating constraints. In addition to reliability, efficiency is also an important concern. It means that the use of optimization process to provide necessary changes or adjustments of controllable components for achieving desired operational and planning objectives while complying reliability requirements. Without the aid of optimization, the power systems may be operated in an uneconomic and inefficient way (e.g. high generation costs and power losses), which unnecessarily costs huge amounts of money. The power system optimization is usually known as Optimal Power Flow (OPF), which has been widely applied to assist power system operation and planning for many years since its first introduction in 1962 by Carpentier [12].

Practical power system operation usually consists of following four stages with distinct time frames [10], [13] and the OPF can be applied to decision-making at any operational time frames:

- **Long-term power system planning**: The OPF at this stage could be used to provide optimal decisions for investment of power plant, expansion of transmission system or distribution network, allocation of power apparatus (e.g. capacitor, reactor, FACTs), and so on, which ensures the power systems are adequate and reliable to supply the future load demand increases.

- **Power system operational scheduling**: In order to meet the daily or hourly variation of load demands and ensure the system always has sufficient generation resources on-line or standby at minimum cost, the system operators need to efficiently schedule the start-up time and shut-down time of generators according to their generation cost curves. Practically, the OPF at this stage is performed day-ahead or hourly ahead to select most economic generation Unit Commitment (UC) and provide advisory optimal setpoints for committed generators to meet the forecasted load at operating day or operating hours. Moreover, the OPF also needs to solve the generation reserves requirement to guarantee that the reserves are sufficient for real-time operation reliably dealing with forecasting errors, contingencies, instantaneous peak-demands, which is a mandatory procedure of power system operation. For example, the operator of New England power system usually maintains reserves between 2185 MW and 2875 MW, which is almost 10 % of its daily peak demand [14]. Furthermore, the reserves in practice are divided into different categories based on their ramp rate and response time, which may be found in [15], [16].
- **Real-Time Economic Dispatch**: At this time frame, the operators need to revise the day-ahead or hourly-ahead operational schedules to ensure the system reliability. Note that the OPF will not make any commitment decisions at this stage. The OPF is implemented five minutes ahead to make dispatching decisions for SCADA controlled dispatchable generators and provide necessary changes for controllable components to meet the projected load demands of next target five minutes interval at minimum total costs while satisfying the system constraints. However, some committed generators from previous operational scheduling stage are non-dispatchable in real-time, such as a nuclear power plant usually supplies the system basis load with constant output for the sake of security, and the bilaterally contracted power plants have their own generation schedules, which are informed to system operators but are not willing to be dispatched by system operators. Similar to operational scheduling process, this stage also solves the minimum cost of reserves to deal with the uncertainties. Moreover, the OPF at this stage could be switched to different forms to achieve desired objectives based on operator's option. For example, the Security Constrained Economic Dispatch (SCED) could be used to provide preventive or corrective control actions against credit contingencies and the OPF could be changed to provide necessary changes to minimize power losses.
- **Automatic control**: This is the automatic control process, which is implemented on a minute to minute or shorter interval by EMS that the system operators do not have sufficient time to balance the generation and load and address the system violations. The OPF is combined with LFC to achieve AGC to correct the imbalance between generation and load within the scheduled real-time economic dispatch interval at minimum cost. This is a regulating procedure and the regulating AGC units are usually designated at operational scheduling or real-time economic dispatch stage. Practically, the LFC detects the imbalance every few

seconds, the OPF at this stage must issue the optimal decisions within a matter of seconds for regulating AGC units. In addition to AGC, the OPF can also be applied to AVC to minimize reactive power dispatch to maintain the voltage within specified limits.

These operational time frames are illustrated in Fig. 1.5 [13], where the latter operational stages fine tune earlier operational decisions using a more accurate power system model with more limited decision space. As shown in Fig. 1.6 [16], the 5 minutes economic dispatch corrects the hourly generation schedule, which is very close to the actual load and the deviations from actual load will be further corrected by AGC. Eventually, the power system is operated in a reliable and efficient manner to continually supply the uncertain load demands.

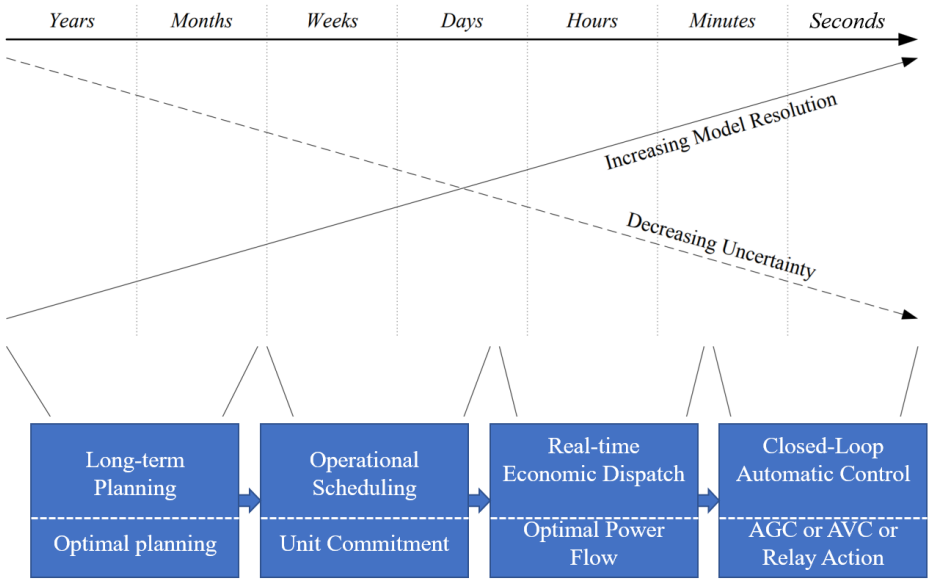


Fig. 1.5 Power system operational time horizon [13]

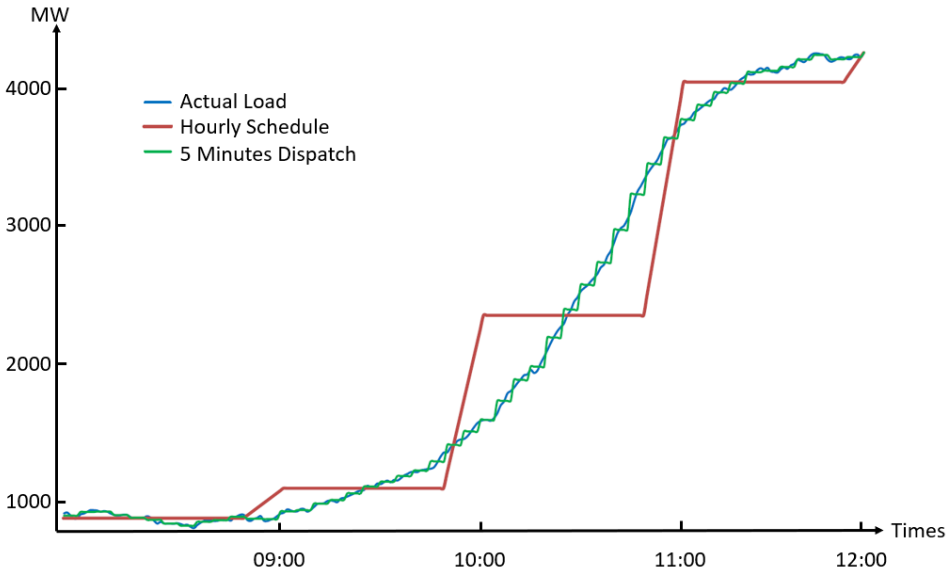


Fig. 1.6 An example of operational scheduling and economic dispatch [16]

1.4 Motivation

As mentioned in previous sections, OPF is the one of vital software tools of EMS for assisting power system operation. It is able to provide different system operation stages with optimal decisions and necessary control strategies to achieve desired operational objectives, while maintaining the system reliability within operational and physical constraints. In practice, OPF is usually performed in the control centre many times a day, as often as every 5 minutes or every few seconds, to let the operators apply the calculated OPF results manually or to apply these results automatically. For example the American power system operation, OPF is usually solved daily in 2 hours, hourly in 15 minutes, each five minutes in 1 minute and for automatic control in the matter of seconds [8]. This requires that the OPF solution must be obtained within reasonable time. If the OPF is solved within impractically long time, the obtained OPF solution will become infeasible or invalid for implementation, especially for the Real-Time (RT) applications. For example, the real-time economic dispatch usually considers the 5 minutes ramp rate (i.e. 5 minutes feasible dispatch range) to re-dispatch generators' active power outputs [16], the generators may be unable to reach the issued optimal outputs if the OPF calculation time is too long. On the other hand, a fast OPF solution technique could allow for corrected optimal decisions to be made if the forecasted errors or contingencies are identified before the implementation of previously obtained optimal decision. Moreover, a fast OPF algorithm could provide system operators with a set of solutions to select the preferable decision. Furthermore, the system operators may expect that the OPF algorithms could provide feasible optimal solution within reasonable time to bring the power system back to normal condition from emergency or alarm condition [10]. Along with the high penetration of renewable generations, electric vehicles and energy storage devices, power systems become more uncertain and complex than ever; in order to maintain the system reliability, the power systems should be adjusted more frequently and the OPF solution techniques need to be fast enough [17], [18].

In addition to the computational efficiency of OPF solution technique, solution quality and power system modelling are also very important. It is accepted that the ultimate goal of power system operation with the aid of OPF is full AC power flow based OPF (ACOPF) [8], [19], [20], because the ACOPF can represent the exact system and worldwide power systems are mainly operated in the AC environment. As mentioned in early section, DC networks are not widespread and it is difficult to assert when DC networks will fully replace the AC power systems, thus this work conservatively considers ACOPF problems. The data from US Energy information Administration (EIA) reported that improvements on accuracy of ACOPF solution and ACOPF formulation have saved huge amounts of generation cost and energy annually [8], [19]. Currently, the approximation OPF formulations are widely employed in practical AC power system operation, because they are relatively easy to solve and they produce reasonable OPF solutions at excellent computational speed [19], [21]. For instance, DC power flow based OPF (DCOPF) holds voltage magnitude constant, ignores reactive power and uses approximated line flow equation or penalty active power losses

formula [22]; decoupled ACOPF assumes that active power–voltage magnitude and reactive power-voltage angle are weakly coupled [23]. Under these assumptions, the ACOPF problems could be approximated to a linear form and thus they could be easily solved by many mature optimization solvers. However, the global optimal solution may not be existing in such approximated OPF model and the deployment of DCOPF or decoupled ACOPF solutions in AC power systems may lead to unwanted system conditions, such as instability or system violations and high generation cost [24]–[26]. Therefore it is a risk and uneconomic way to use the simplified OPF models, especially under stressed system conditions, if their accuracy in the particular system are not carefully verified [27]. On the other hand, in order to maintain system reliability and improve system efficiency, the closer to real-time operation, the more important the power system modelling accuracy tend to be (e.g. full ACOPF) [13], [28]; but the detailed system modelling increases the difficulty of solving OPF problems. All these factors require that the ACOPF (latter the term OPF means ACOPF unless indicated) solution techniques should be accurate and fast enough, producing global optimal solution within very short CPU time to maintain the system reliability and efficiency. It is true that a more accurate and more efficient OPF solution technique is always welcome as it could greatly improve the reliability and efficiency of power systems [20], [28]–[30].

OPF mathematically leads to a large-scale, nonlinear, nonconvex and highly constrained optimization problem where the continuous and discrete variables are optimized within specified constraints to minimize or maximize the desired operational objective function [8]. However, the reality is that 50 years after the OPF problem was first proposed, it still lacks a robust, accurate and fast algorithm for solving OPF problems [30]–[35]; while the ongoing increase of power system size and the growing use of both renewable sources and advanced power electronic technologies, on the other hand, continuously increase the complexity of OPF problems over the years [18], [20]. This has led the U.S. Department of Energy and the IEEE PES Working Group on Modern Heuristic Optimization (WGMHO) to initiate power system optimization competitions in recent years [36], [37] to advance the development of OPF solution techniques. The main aim is to find the accurate and effective OPF algorithms to solve the increasingly complex OPF problems and thus improve the reliability and the efficiency of power systems associated with existing and emerging challenges. Current OPF algorithms could be generally divided into two classifications: deterministic methods and evolutionary computation (also known as stochastic or heuristic optimization). Before twenty-first century, the deterministic methods – like Simplex Method [38], Sequential Linear Programming Method [39], Nonlinear Programming Method [40], Newton based Method [41] and Interior Point Method [42], have been applied to solve practical OPF problems for many years owing to their excellent computational speed. As demonstrated in [43], the deterministic methods can produce reasonable OPF solution within a minute for large-scale power systems (e.g. 3120-bus test power system). These algorithms are approximation techniques that simplify the OPF problem to a linear, continuous or unconstrained form (i.e. depend on the selection of optimization solver) in order to obtain the reasonably acceptable results [30], [32], [34]. The major drawback is that they do not

guarantee the recovery of global optimum or even feasible solution, because they rely strongly on the convexity problem (while OPF problems are naturally nonconvex) and the choice of an initial start point in the search for an optimum (i.e. local search) [30], [43]. Nonetheless, these methods are very promising for solving DCOPF, the DC power flow constraints of which are linear and no further linearization is required [44]. More recently, convex optimization techniques (e.g. Semidefinite Programming [45], Second-Order Cone Programming [46]) have attracted significant attentions in that, the global optimum of the nonconvex OPF problem can be recovered from the optimal solution of the exact convex relaxation of original OPF under certain sufficient conditions [35], [47], [48]. Although the convex relaxation satisfying certain sufficient conditions is exact and can guarantee recovery of global optimum of some OPF problems [49]–[55], these sufficient conditions may not be always satisfied in practical power systems and thus it may result in inexact relaxations, providing local optimum or even infeasible solutions [47], [48], [56]–[58]. In addition to the exactness issue, the computational speed of relaxation techniques are significantly slower than the mature approximation methods (especially for large-scale OPF problems) due to the extensive number of variables [47], [59], [60]. The mathematical associated drawbacks of deterministic algorithms currently cannot be fully addressed or may be impossible to be addressed. Applying the OPF solutions obtained from current deterministic algorithms to practical power systems may cause the system operating in a low efficiency and unreliable manner that may unnecessarily cost billions of dollars per year and result in waste of energy [8], [32]. For these reasons, the approximation and the relaxation techniques are not studied in this work.

As the competitive alternative to the deterministic methods, the derivative-free evolutionary computation – such as Genetic Algorithm (GA) [61], Particle Swarm Optimization (PSO) [62], [63], Simulated Annealing [64], Artificial Bee Colony [65], Differential Evolution Algorithm [66], [67] and Hybrid Evolutionary Algorithms [68], [69], which are capable to address shortcoming of deterministic methods (e.g. local optimum issue and difficulty in handling discrete variables). Unlike the deterministic methods solely based on local search, the evolutionary computation has additional global search capability, which allows to find the global optimum if sufficient computation time is given [31]. However, this kind of intractable random search techniques is computational inefficient, as it usually requires large number of objective function evaluations (i.e. full AC power flow analysis), yielding impractically long execution time for OPF problems. With today's computer power, they would take few minutes to dozens of minutes to obtain the solution for a 118 bus test power system [70], [71], whereas the deterministic methods only need few seconds [43]. Nevertheless, a recent work [72] reported an overturning story, which has pointed out that the evolutionary computation methods are more reliable and efficient for meeting the needs of future power system from long-term perspective; and the deterministic algorithms may no longer have excellent computational speed when dealing with future smart grid OPF problems, because extensive number of variables and constraints need to be considered. The study [72] has shown that the GAMS optimization software took about 19 hours to deal with a 33-bus network OPF case considering

49,898 variables, whereas the tested evolutionary algorithms achieved better OPF solutions in30 minutes, on an average. Even though many research works have proven that the evolutionary based algorithms have good solution quality, flexibility and scalability when solving various OPF problems, the required running times actually have limited their application in practice, especially for real-time OPF problems considered in current power system operation. This has strongly motivated this work to further improve the performance of evolutionary based algorithms already developed and thus advance them to practical applications as well as enhance power systems reliability and efficiency.

1.5 Aim and Objectives

The significant importance of OPF is introduced and discussed in previous sections. It is found that a more accurate and efficient OPF algorithm is always needed in practice for improving system reliability and saving operational cost. Throughout the literature, current OPF solution techniques suffer from either the solution quality or the computational efficiency issue with varying degrees, which means any current optimization algorithm used to solve practical OPF problems may be considered as a compromise between solution quality and computational efficiency. The solution quality issue of the computationally efficient determinism algorithms is recognized as a very difficult mathematical task to address, whereas the computational efficiency concern of the accurate evolutionary based algorithms could be alleviated with the help of surrogate modelling techniques (i.e. Surrogate Assisted Evolution Computation) [73]. By properly managing the surrogate model in evolutionary computation, the fitness of some individuals, populations, or generations is evaluated using “expensive” real objective function (i.e. full AC power flow analysis in this work), while others are estimated by a “cheap” surrogate model; therefore the necessary number of real objective function evaluations for finding global optimum can be reduced and the computational efficiency is improved. This is considered as the main motivation behind this work, and this work aims to develop a surrogate technique, which could further advance the evolutionary computation to the practical OPF applications, to improve the solution quality and especially the computational efficiency of evolutionary based algorithms when solving OPF problems. Although the surrogate techniques are somewhat new to the field of power system optimization, they have shown significant improvements on the computational efficiency when applied to other optimization problems (e.g. electromagnetic and aerodynamic designs) [73]–[76]. Many surrogate techniques may be found in literature, whilst ‘Kriging’ has been shown to be particularly robust and accurate in capturing the globally nonlinear behaviour of numerical problems [77], [78] and electromagnetic design optimization (previous works reported by the Southampton group [79], [80]). Therefore, this work focuses on exploiting the Kriging surrogate technique to improve the performance of evolutionary computation associated with solving OPF problems. The objectives of this work are listed as follows:

- Review the optimal power flow including the formulations and the solution techniques, which identifies and discusses the potential research gaps.

- Review and critically assess the surrogate assisted evolutionary computation. Since the surrogate techniques have not been applied to solve OPF problems, it is necessary to discuss and summarize their features, which would be greatly helpful for developing the novel algorithms needed for this work.

- Propose the effective method for alleviating the computational burden of Kriging associated with large-scale OPF problems. The application of Kriging has so far been limited to relatively small-scale optimization tasks, because of the need to invert a large correlation matrix with associated O ($n^3 d$) computational cost and memory storage requirements [76], where n is the sample size and d is the dimension size. Thus it is essential to deal with this issue, which ensures the Kriging being a successful and reliable technique for improving the performance of evolutionary based algorithm when solving OPF problems.

- Propose the adaptive approach to balance exploitation and exploration when using Kriging to select the infill point. The balance between exploitation and exploration is very significant for any stochastic based algorithm, because it directly influences the capability of finding the global optimum and the algorithm's overall computation efficiency. If the emphasis is on exploitation, it may be helpful to fast converge to optimal solution. However it may be a risk of finding a local optimum, especially for the multimodal nonlinear optimization problems. On the other hand, if placing too much emphasis on exploration, the overall optimization process will be slowed down although it is likely to find a global optimum. Therefore, it is very important to propose an approach that could adaptively balance the exploitation and exploration during the optimization process, enabling the developed Kriging assisted evolutionary algorithm to find global optimum using minimum computational efforts.

- Propose the efficient strategy for managing the Kriging in the evolutionary based algorithm, as it could be helpful to reduce the total number of objective function evaluations needed for the algorithm to find the global optimum, and thus improve the computational efficiency of evolutionary based algorithm. Hence, an efficient strategy needs to be proposed to guide the algorithm how to use the real fitness and the estimated fitness effectively and alternatively.

- Propose the Kriging assisted evolutionary based algorithms and test their performance on benchmark optimization functions and the IEEE test power systems. In order to verify the applicability and reliability of proposed Kriging strategies applying to evolutionary based

algorithms, a combination of proposed Kriging strategies with GA and PSO is the novel algorithms here, because GA and PSO amongst various evolutionary based algorithms are best known with proven performance and simple coding structures.

The original contributions include:

- Introduction of the novel idea and surrogate modelling technique to improve the performance of evolutionary based algorithms when solving OPF problems.

- The simple and effective method is proposed to alleviate the computational burden of Kriging estimation to an acceptable and manageable level when solving large-scale problem.

- The adaptive approach used for balancing the exploration and exploration during the optimization process is proposed.

- The efficient strategy used for managing Kriging in evolutionary based algorithms is proposed.

- By applying the proposed Kriging strategies to GA and PSO, the computational efficiency of original GA and PSO is enhanced at least 50% when solving the test OPF problems, while their OPF solution quality is improved.

1.6 Outline

Chapter 2 critically reviews the OPF formulations and solution techniques.

Chapter 3 discusses the fundamental of surrogate methods including Polynomial Regression, Radial Basis Function and Kriging for developing the proposed algorithms.

Chapter 4 reviews and studies the surrogate assisted optimization techniques to identify the potential one for OPF problems.

Chapter 5 develops and tests the individual-based Kriging assisted algorithms.

Chapter 6 develops and tests the pre-selection Kriging assisted algorithms.

Chapter 7 summarises the work and suggests the potential work for further research.

Chapter 2 Overview of Optimal Power Flow

2.1 Introduction

Modern power system operation is an onerous task, which usually requires various decision-making support tools to secure system reliability and efficiency. The Optimal Power Flow, first formulated by Carpentier in 1962 [12], has been used as an essential tool in planning, operation, and dealing with energy management aspects of power system. For long-term planning purposes, OPF could provide optimal decisions to invest and expand the power system infrastructures (e.g. power plant, substation, transmission grid, etc.) to meet the future demands. For daily power system operations, the OPF could be used in following operational stages [10], [27], [81], [82]:

- **Day-Ahead Unit Commitment (DAUC)**: The OPF is implemented day-ahead to select the most economic generators from all participated generators (and schedule their level of output) to meet the forecasted demands at every hour interval of the following day while satisfying the specified constraints.
- **Real-Time Economic Dispatch (RTED)**: The OPF is implemented every 5-15 minutes (depends on different systems) ahead to dispatch the dispatchable generators at minimum generation cost to meet the estimated load of next 5-15 minutes interval while maintaining the system reliability.
- **Automatic Generation Control**: The OPF under normal system state may be implemented every few seconds (e.g. 6 second) to regulate the regulable generators at minimum cost to maintain the system frequency within specified level.

The above operational stages usually associate with Day-Ahead, Real-Time and Ancillary Service markets, respectively, because power system operators may also be responsible for operating these forward electricity market [81]–[85], as shown in Fig. 2.1. Practically, the energy suppliers and the load serving utilities submit their generation offers and load bids to the electricity market information

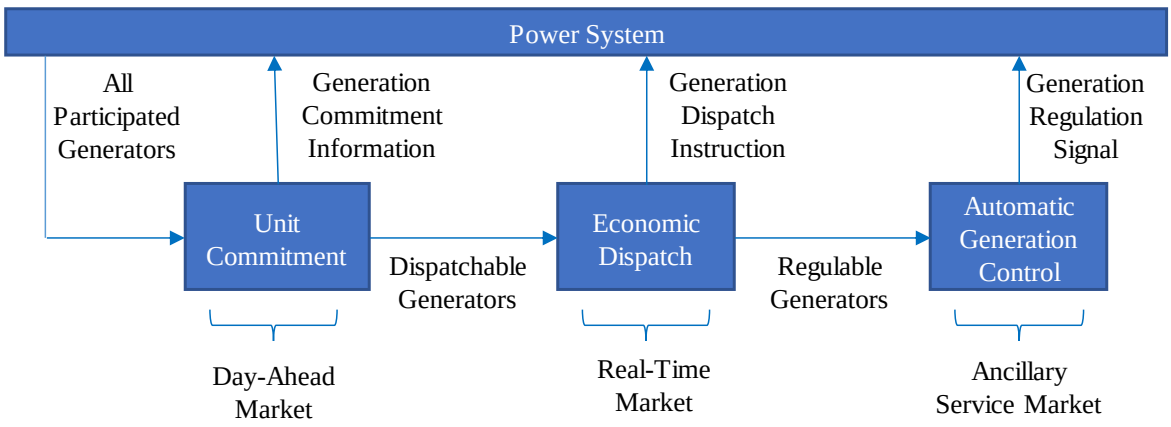


Fig. 2.1 Power system operations and electricity markets

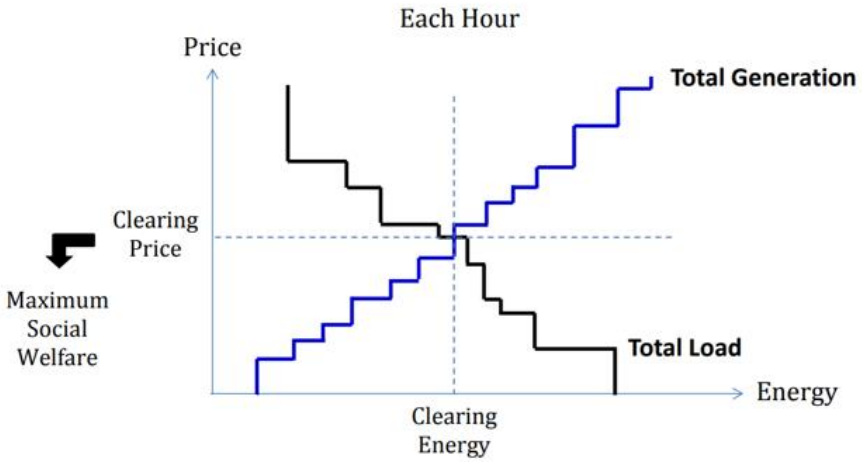


Fig. 2.2 Electricity market clearing

system (i.e. an interface between power system operators and participants), respectively. Associating the power system operation with the electricity market, the generation and the demand are always balanced in an active manner, ensuring sufficient online and standby generation capacities at any stage. The OPF used in power system and electricity market operations could maximize the social welfare, providing market clearing prices, market clearing energy and market clearing reserves [84], as illustrated in Fig. 2.2.

Over the years OPF problems have been extensively studied in the literature and various solution techniques have been proposed to solve these nonlinear and nonconvex optimization problems [30], [31], [35], [86]. The robust and efficient OPF formulation and solution technique can be able to generate better electricity market signal and greatly improve the power system reliability and efficiency [8], [87]. The complex electricity market topic is out of scope of the thesis; however, for who interested about it, more details may be found in [81]–[85]. Therefore, this chapter is focused on overview of OPF problems and critical review of OPF solution techniques, which are helpful to better understand OPF and address the associated challenges.

2.2 OPF Problems

A variety of problems have been formulated and named as OPF to find optimal solutions to various objective functions subject to different variables and constraints. In general, OPF is recognized as being nonlinear, nonconvex and large-scale optimization problem containing both continuous and discrete control variables [8]. The global optimum of such optimization problem is very difficult to be found, because many local optimal traps exist, as shown in Fig. 2.3. Nevertheless, the difficulties of solving OPF problems depend critically on the selection of objective functions, variables and constraints because they influence the accuracy of power system representation and the presence of nonlinearity and nonconvexity. This section mainly gives an overview of OPF problems.

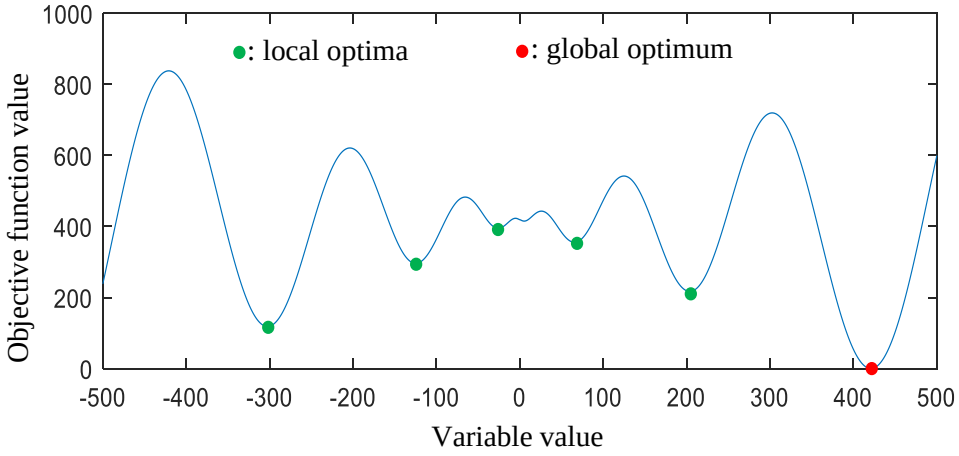


Fig. 2.3 Nonlinear and nonconvex optimization problems with many local optima

2.2.1 General OPF Formulation

OPF could be expressed in a general way like any other constrained optimization problems as [30]:

$$\text{min. } f(\boldsymbol{x}, \boldsymbol{u}) \tag{2.1}$$

$$\text{s. t. } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = 0 \tag{2.2}$$

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \tag{2.3}$$

where $f(\boldsymbol{x}, \boldsymbol{u})$ is the objective function to be minimized by adjusting control variables $\boldsymbol{u}$ while satisfying the equality constraints $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$ and inequality constraints $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$, and $\boldsymbol{x}$ denotes the vector of state variables. Depending on the selection of (2.1) - (2.3), the OPF may become a linear, nonlinear, nonconvex, or mixed-integer programming problem. For the classical OPF formulation, the control variables include active power generations and voltage magnitudes at generator buses (i.e. PV buses), tap change of all transformers, and shunt compensators; while the state variables include active and reactive power generations at slack bus, reactive power generations at generator buses, voltage magnitudes and voltage phase angles at load buses (i.e. PQ buses), and transmission line flows. Moreover, the AC power flow equations and the limits on aforementioned variables (state and control variables) are considered as the equality and inequality constrains, respectively, for classical OPF formulation. The following subsections introduce and discuss the objective functions, the variables and the constraints, which could be used to formulate different OPF problems.

2.2.2 Objective functions

The most common OPF objective function used in practice is the minimization of generation costs with consideration of a set of constraints. For example, the DAUC (i.e. dynamic OPF) is solved day-ahead or few hours ahead to minimize the generation costs across multiple time periods via

scheduling on-off status and output level of each participated generator [88]; the RTED (i.e. static OPF) is solved in a real-time or near real-time to dispatch each dispatchable generator at a certain time of interest to minimize the generation cost [13]. Moreover, the generation costs functions are usually nonlinear and they may be expressed by quadratic cost curves [89], piecewise linear curves [90], or piecewise quadratic curves [91]. In addition to minimization of generation cost, many other OPF objective functions have been proposed to deal with the particular operational concerns, such as minimization of active power losses [92] (for improving system efficiency) and minimization of voltage deviation (for improving power quality) [93]. Referred to [30], [94]–[96], more OPF objective functions may be found and it is noticed that these objective functions are of being nonlinearity. Therefore the nonlinear objective function is the one of main issue resulting in the difficult of solving OPF problems.

2.2.3 **Variables**

Normally, OPF problems have two types of variables: control variables and state variables. The optimal values of control variables must be determinate to calculate the state variables as well as constraints and objective function value. The classical OPF formulation uses all available variables to represent the exact power system, while complying Kirchhoff's laws and operational constraints. In addition to the classical control variables, the loads could be controlled for optimizing demand response problem [97], the HVDC link MW flows may be optimized when dealing with the hybrid AC-DC OPF [98] problem, and FACTS controls are helpful to efficiently manage the power system [99]. Besides, the OPF (e.g. DAUC and topology OPF [100]) may be formulated as a Mixed-Integer Nonlinear Programming (MINLP) problem, since control variables in practice may be continuous, discrete (e.g. stepwise tap change) or binary (e.g. switchable devices), which usually leads to a discontinuous solution space. The MINLP remains challenging for current deterministic algorithms and thus it is not uncommon to use evolutionary based algorithms to deal with such problems [47], [101].

Overall, the selection of control variables usually differs among OPF problems according to the particular concerns. However, it should be careful with the effectiveness of control variables, because the power system may undergo uneconomic and unwanted operating state if the defective or ineffective controls are used to optimize an objective function and enforce the constraints [102]–[104]. As indicated in [28], the use of both effective active power controls and defective voltage magnitude and reactive power controls to minimize generation cost could easily lead to the power system with large reactive power flows and poor voltage profiles; despite each control variable contributes to objective function in a varying degree and optimizing all control variables would produce lower generation cost. This perhaps one of main reason why DCOPF and decoupled ACOPF still being widely used in practically daily power system operations, as voltage magnitude and reactive power controls have insignificant effects on generation cost [19]. Moreover, the active power

controls should not be controlled to minimize the total power system losses (i.e. Optimal Reactive Power Flow), as they would lead to high generation cost that is inconsistent with economic principles [8], [27]. On the other hand, the elimination of insignificant and ineffectiveness controls is helpful to alleviate the computational challenge of OPF, as the number of control variables directly affects OPF solution space and difficulty (e.g. n-control variables means n-dimensional optimization problem).

2.2.4 **Constraints**

In order to provide the reliable and feasible OPF solution, the constraints must be imposed to the state and control variables and other operational factors. Commonly, the constraints considered in OPF formulation can be categorized into equality and inequality constraints. To accurately model the power system state, OPF formulation usually considers power flow equations as the equality constraints. Reader may referee to [27], [42] for more details about power flow analysis (or load flow analysis). However, the studies [48], [105], [106] have indicated that the use of nonlinear AC power flow equations (i.e. full AC network model) in the classical OPF formulation introduce nonconvexity to the OPF solution space, while the current deterministic algorithms used to solve such nonconvex OPF problem cannot guarantee the convergence and the global optimum [35]. To ensure that the reasonable OPF solutions can be found within an acceptable time, the practical applications usually adopt a linearized network model (e.g. DC model or decoupled AC power flow) instead [19], [30], [47], sacrificing the accuracy of OPF formulation but improving the computational efficiency. In this way, the OPF formulation ignores system losses and reactive power and it assumes a flat voltage profile, therefore the relationship between active power and voltage phase angle is linear [22], [23], [107]. It is worthy to mention that the OPF used in DC network is also a nonconvex optimization problem, because the power flow equations of DC networks associated with the presence of power converters is nonlinear [108], [109]. Although the OPF with linearized network model attempts to obtain the optimal solution as close as to the original full ACOPF, the obtained OPF solutions may be uneconomic or even insecure, especially for the heavily loaded systems and the systems with strong coupling between active and reactive power [110]. For example, [111] demonstrated that the accuracy of DCOPF is acceptable if the system voltage profile tends to be flat, the system is not stressed, and the R/X ratio of transmission line is less than 0.25; the DCOPF model with estimated losses could be accurate for modelling the active power flow, but very large errors can occur for certain branches in the power systems [107]; [24], [26] studied that the DCOPF model assuming flat voltage profile can lead to significant error, because the actual AC power flow solution have large differences in voltage in different buses of the tested systems. Therefore, the accuracy of linearized OPF models must be carefully evaluated for different systems and the additional information of the power system is required for linearized OPF, while further improvements on the accuracy of linearized OPF models is needed to improve the system reliability and efficiency. From operational point of view, controlling the reactive power and voltage magnitude is an efficient way

to limit the dispatch of active power when the power system is approaching its transmission limit [112]. Hence, an accurate modelling of losses, reactive power and voltage magnitude in OPF formulation is necessary, while the full ACOPF may be the idea alternative. Notwithstanding, the major issue is how to solve the ACOPF efficiently, and the thesis is to improve the computational performance of OPF solver when solving ACOPF problems, rather than finding an accurate linearized OPF model.

In addition to the equality constraints, OPF problems also need to consider the inequality constraints, which usually include the limits (maximum or minimum or both) on the all relevant state and control variables. The limits of variables could be divided into two types: physical limits and operational limits. The physical limits in principle cannot be violated (e.g. generation output, tap change), whereas the operational limits depending on the grid code can be violated temporarily (e.g. thermal limit of transmission line [28]). By using different constraints, various OPF formulates could also be formulated. For example, the classic OPF formulation first proposed by [12] is an extension of Economic Dispatch: the ED only dispatches the generator output with or without system losses to minimize the total generation cost, while classical OPF optimizes all power flow within the system limits to minimize generation cost. In practice, the OPF may be required to consider both base case and post-contingency state constraints and such problem is usually termed as Security-Constrained OPF (SCOPF), which is able to provide preventive or corrective optimal solutions to deal with system outages and failures, as introduced and discussed in [27], [28]. In order to ensure that the system can withstand contingencies, the Transient Stability-Constrained OPF (TSCOPF) has been proposed by considering additional transient or dynamic stability constrains [113]. Moreover, the Stochastic OPF [114] and Probabilistic OPF [115] have been proposed to deal with the system uncertainties, arising from loads, renewable generations, and other uncertain factors, by regarding the uncertainties as a part of constraints. Overall, the OPF constraints are needed to establish a feasible solution region, where it may be nonlinear or even nonconvex depending on the selected constraints and any OPF solution within it is feasible and reliable.

2.2.5 **Summary**

This section gives a general overview of OPF to introduce and discuss the source of difficulties (i.e. nonlinearity, nonconvexity and discontinuity) of solving this problem. The classical OPF, extended to various formulations to address different engineering concerns by using different objective functions, variables and constraints, has been introduced. It is found that many works focused on the development of simple OPF formulations with approximated network model to reduce the nonlinearity and nonconvexity, however, solving such approximated OPF problems usually come up with inaccurately optimal solution or infeasible optimal solution, putting the power system in a dangerously operating condition. It is also noticed that the classical OPF (i.e. full AC OPF model) is nonlinear and nonconvex with many local optimal solutions, so these natures in principle also exist

in the extensions (not include DCOPF). Therefore, this thesis is mainly focused on the classical OPF without the consideration of security and transient constraints and other special variables (e.g. stochastic loads, energy storage devices, etc.). The detailed OPF formulations considered in this work are introduced in later chapter.

2.3 OPF Solution Techniques

In the OPF literature, there are also many research works focused on the development of robust and fast optimization algorithm for solving the OPF, except for finding the accurately and reliably simplified OPF model to replace the complexly nonlinear and nonconvex OPF problem. The existing OPF solution techniques may be classified into two categories – including deterministic methods and evolutionary computation. In fact, the development of OPF solution techniques is strongly connected with the development of OPF formulations. The particular choice of objective functions, variables and constraints could lead the OPF problem to different forms – like Linear Programming (LP) [39], Quadratic Programming (QP) [116], Nonlinear Programming (NLP) [12] Mixed-Integer Nonlinear Programming (MINLP) [117], Second Order Cone Programming (SOCP) [118] and Semidefinite Programming (SDP) [45], while the selection of the programming formulation usually implicates both solution technique and solution accuracy. In other words, to use the deterministic methods while assuring their convergence performance, an appropriate OPF programming formulation must be defined first [30]. In practice, the deterministic methods – such as Simplex Method (SM) [38] and Sequential Linear Programming Method (SLP) [39], Sequential Quadratic Programming Method (SQP) [119], Newton based Method (NM) [41] and Interior Point Method (IPM) [42], have been applied for many years to solve the OPF problems in the form of LP, QP and NLP (i.e. approximation techniques), respectively. Because if the OPF problem defined by the approximated form is solved using the corresponding deterministic method, the reasonable OPF solution could be obtained within the acceptable time, meeting the power industry requirement [32]. In the literature, the SOCP and SDP (i.e. relaxation techniques) have gained lots of attentions in recent years due to their capability of finding global optimum, but they are currently limited to the certain types of OPF problem [51]. Even though the nonconvex OPF problem could be tailored to a solvable convexity problem, the presence of approximation or relaxation error can cause that the optimal point found in such approximated or relaxed problem differs from the desired one in original OPF problem [48]. Moreover, the deterministic methods usually consider the OPF problem as being continuous [30], [120], thus the they cannot properly handle the discrete variables, providing the sub-optimal solution. Consequently, the deterministic methods cannot guarantee to find the desired global optimum and their OPF solution quality is a concern. In order to overcome the drawback of deterministic methods, the evolutionary computation has been proposed as a very competitive alternative. Due to the evolutionary based algorithms can solve the full OPF problem in the accurate form of MINLP

without losing any power system modelling information, while they can easily handle the discrete variables [31], [47]. However, the evolutionary based algorithms usually require impractical computation time to find the global optimum, limiting their application in practice, especially in Real-Time applications. The following sub-sections review and discuss the pros and cons of state-of-the-art OPF solution techniques in detail.

2.3.1 Deterministic Methods

The optimization of deterministic methods is like a pure exploitation (local search) process that refines a start point within a nearby region to improve the solution. Such optimization process begins with a pre-defined solution point (i.e. initial start point), and then the local gradient is calculated through derivative of objective function and constraints to suggest a search direction to update the solution [32]. After few times performing along the searching direction, the solution would eventually change from the initial point to a global or local optimum. This may be the reason why the deterministic methods have cheap computational cost and fast convergent speed, so they have been widely applied in industry and commercial software. However, the selection of initial point is extremely important for the derivative based deterministic methods to find out the global optimum [121], [122]. If the optimization problem is nonconvex with multiple local optima, an inappropriate start point could easily lead the deterministic methods converge to a local optimum, while a feasible start point is not easy to be defined in practice [43]. In addition, the deterministic methods cannot easily deal with the discrete variables and the non-derivative problems, since the final solution is arrived by the derivative gradient information. Hence, the use of deterministic method for solving OPF problems could be deemed as a compromise between the computational cost and the solution quality. Some common deterministic OPF methods are introduced and discussed as follows.

2.3.1.1 Linear Programming

The key of using LP methods for solving nonlinear OPF problem is that converting the OPF to an approximated linear form although it is natively nonlinear. In order to linearize the OPF, its inequality constraints should be efficiently handled as the equality constraints to give a linear system (i.e. linear equations) and a feasible solution region. For example, in [39], a Successive Linear Programming for OPF was proposed and the standard equations given in (2.1) – (2.3) could be linearized by first-order Taylor's expansion as

$$min.\ f(\boldsymbol{x}^t, \boldsymbol{u}^t) + \nabla_{\boldsymbol{x}}^T f(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{x} + \nabla_{\boldsymbol{u}}^T f(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{u} \tag{2.4}$$

$$s.t.\ \boldsymbol{g}(\boldsymbol{x}^t, \boldsymbol{u}^t) + \nabla_{\boldsymbol{x}}^T \boldsymbol{g}(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{x} + \nabla_{\boldsymbol{u}}^T \boldsymbol{g}(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{u} = 0 \tag{2.5}$$

$$\boldsymbol{h}(\boldsymbol{x}^t, \boldsymbol{u}^t) + \nabla_{\boldsymbol{x}}^T \boldsymbol{h}(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{x} + \nabla_{\boldsymbol{u}}^T \boldsymbol{h}(\boldsymbol{x}^t, \boldsymbol{u}^t)\Delta\boldsymbol{u} \geq 0, \tag{2.6}$$

where t is the current iteration number, T and ∇ is the transpose operator and the derivative operator, respectively, $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{u}$ are the step length vectors to be calculated to update the current solution to

an unknown solution point (i.e. $\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} + \Delta\boldsymbol{x}$ and $\boldsymbol{u}^{t+1} = \boldsymbol{u}^{t} + \Delta\boldsymbol{u}$). It can be seen from above the LP is approximately and locally constructed as the incremental of the current solution and the solution is iteratively updated by the change in variables until the algorithm is convergent (e.g. $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{u}$ are smaller than a specified value). The relationships between $\boldsymbol{x}$ and $\boldsymbol{u}$ may be expressed by extending the Newton-Raphson power flow equation as the incrementally linearized network

$$\Delta\boldsymbol{S} = \boldsymbol{J}\Delta\boldsymbol{x} + \boldsymbol{A}\Delta\boldsymbol{P}_{\boldsymbol{g}} + \boldsymbol{B}\Delta\boldsymbol{V}_{\boldsymbol{g}} + \boldsymbol{C}\Delta\boldsymbol{T} + \boldsymbol{D}\Delta\boldsymbol{Q}_{\boldsymbol{c}} \tag{2.7}$$

$$\Delta\boldsymbol{x} = \boldsymbol{J}^{-1}\left[\Delta\boldsymbol{S} - \boldsymbol{A}\Delta\boldsymbol{P}_{\boldsymbol{g}} - \boldsymbol{B}\boldsymbol{V}_{\boldsymbol{g}} - \boldsymbol{C}\Delta\boldsymbol{T} - \boldsymbol{D}\Delta\boldsymbol{Q}_{\boldsymbol{c}}\right], \tag{2.8}$$

where $\Delta\boldsymbol{S}$ is the mismatch vector of active and reactive power at PV and PQ buses, $\boldsymbol{J}$ is the Jacobian matrix, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are the sensitivity matrices with respect to active outputs ($\boldsymbol{P}_{\boldsymbol{g}}$), generator voltage magnitude ($\boldsymbol{V}_{\boldsymbol{g}}$), tap changes ($\boldsymbol{T}$) and shunt compensators ($\boldsymbol{Q}_{\boldsymbol{sh}}$), respectively. By substituting (2.8) into (2.4) – (2.6), the problem becomes a linear problem depending only on the control variables. In addition, the constraints of changing in state variables are expressed as the linear function of control variables like (2.8). This would allow the well-developed LP method, such as SM and IPM, to solve OPF problems in the form of LP efficiently. Due to the LP normally gives a convex solution region, the SM with the aid of slackness conditions (for converting the inequalities to equalities), tableaus and pivoting rule is able to search the optimal solution from one vertex to another along the edges, until the change in step length is sufficient small. However, it may require exponential time in the worst case that the SM visits all 2^n extreme points [123], [124], where n is the number of control variables. Moreover, if the pivoting operation is not properly used then the "cycling" problem may occur, which the SM may search among few vertices without improving the solution [125].

To avoid the "cycling" problem, the IPM was first proposed in [126] to deal with LP problems more robustly and efficiently. First, the (2.4) – (2.6) could be simply expressed as the standard LP form

$$min.\ \boldsymbol{c}^{T}\boldsymbol{x} \tag{2.9}$$

$$s.\,t.\ \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.10}$$

$$\boldsymbol{x} \geq 0, \tag{2.11}$$

where $\boldsymbol{c}$ and $\boldsymbol{b}$ are the coefficient vectors, $\boldsymbol{A}$ is an $m * n$ coefficient matrix. These formulations are usually called primal problem, while the associated dual problem could be formulated as

$$max.\ \boldsymbol{b}^{T}\boldsymbol{y} \tag{2.12}$$

$$s.\,t.\ \ \boldsymbol{A}^{T}\boldsymbol{y} \leq \boldsymbol{c}. \tag{2.13}$$

The dual problem can be seen as the transpose of primal problem. Ideally, the optimal value of $\boldsymbol{c}^T\boldsymbol{x}$ must be equal with the optimal value of $\boldsymbol{b}^T\boldsymbol{y}$. To simultaneously find out the optimal solutions of $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ using IPM, the following optimality conditions must be satisfied

- Primal feasibility: $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$
- Dual feasibility: $\boldsymbol{A}^T\boldsymbol{y}^* + \boldsymbol{s}^* = \boldsymbol{c}$
- Complementarity: $\boldsymbol{x}_i^*\boldsymbol{s}_i^* = \mu$ (for each variable $i = 1, \dots, n$)

The dual feasibility is converted from inequality constraint (2.13) by adding the nonnegative slack variables $\boldsymbol{s}$. It is called Affine Scaling IPM if complementarity $\boldsymbol{x}_i^*\boldsymbol{s}_i^*$ directly set to zero, while it is called Centre Barrier IPM if complementarity $\boldsymbol{x}_i^*\boldsymbol{s}_i^*$ directly set to μ [127]. By multiplying $\boldsymbol{x}^*$ on both sides of the dual feasibility equation, it could be expressed by $\boldsymbol{s}^T\boldsymbol{x}^* = (\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y}^*)^T\boldsymbol{x}^* = \boldsymbol{c}^T\boldsymbol{x}^* - \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{c}^T\boldsymbol{x}^* - \boldsymbol{b}^T\boldsymbol{y}^*$ as the measure of duality gap. According to primal-dual theorem [128], the optimal solutions of primal and dual problems are the same only if the duality gap is equal to zero (i.e. strong duality, otherwise weak duality). Therefore the complementarity condition $\boldsymbol{x}_i^*\boldsymbol{s}_i^* = \mu$ is necessary as it could be used to measure the duality gap or the solution quality. To solve this primal-dual optimization problem, the optimality conditions may be rewritten as linear Newton equations satisfying Karush–Kuhn–Tucker (KKT) condition

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^T & \boldsymbol{I} \\ \boldsymbol{S} & 0 & \boldsymbol{X} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} - \boldsymbol{s} \\ \mu - \boldsymbol{X}\boldsymbol{S} \end{bmatrix}, \tag{2.14}$$

where $\boldsymbol{S}$, $\boldsymbol{X}$ and $\boldsymbol{I}$ are the diagonal matrices with diagonal terms $\boldsymbol{s}$, $\boldsymbol{x}$ and 1, respectively. The variables are updated iteratively as $(\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1}, \boldsymbol{s}^{t+1}) = (\boldsymbol{x}^t, \boldsymbol{y}^t, \boldsymbol{s}^t) + \boldsymbol{\alpha}(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{s})$ until the values of right-hand side of (2.14) approach the specified tolerance, where $\boldsymbol{\alpha}$ is the step length coefficient between 0 and 1 which should be fixed or adjusted during the optimization to improve the IPMs' convergent performance [128]. Moreover, the above optimality and KKT condition can also be derived through the partial derivative of the Lagrangian function of the primal problem with respect to each variable [125]

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) = \boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{\mu}\sum_{i=1}^{n}\log(\boldsymbol{x}_i) - \boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{2.15}$$

where the $\boldsymbol{\mu}$ and $\boldsymbol{y}$ here are considered as Lagrange multiplier for the primal problem. It is noticed that the originally constrained optimization problem is converted to a unconstrainted problem, while the second term named logarithmic barrier function is important to ensure that the feasible solution could be found. Go back to the primal problem, (2.11) is replaced by the logarithmic barrier function, therefore the term $\log(\boldsymbol{x}_i)$ as well as the Lagrangian function tend to infinity if $\boldsymbol{x}_i$ approaches to zero. In this way, it could keep the search path away from the boundary unless the solution begins to converge to the optimum, as it has been proven in IPMs that the search path from the start point to

final optimum may be acting like an arc (named central path) [129]. This could potentially alleviate the drawback of SM, which searches along the edges that may require exponential time in the worst case [42].

These simple, robust and fast LP methods seem to be attractive; however, they may provide local optimum or even infeasible solution for nonconvex OPF problems, due to the fact: their locally searching nature, the LP is locally constructed around the start point [30], and the duality gap cannot be always guaranteed to zero or nearly zero [49]. Furthermore, the presence of degeneracy (primal or dual) may cause redundant work (e.g. cycling), significantly slowing down the computational efficiency of LP methods [130], [131]. Nonetheless, it has been discussed in [44], [132] that the LP methods can be directly applied to solve DCOPF, since the DC power flow equations are a fully linear set, no further linearization is required. In this case, the optimal solution of linear DCOPF problem could be yielded in one iteration (e.g. applying LP method to solve DCOPF is non-iterative), therefore the application of using LP methods to solve DCOPF are quite common in power industry.

2.3.1.2 **Gradient based method**

As introduced in previous sub-section, to use the LP methods, the nonlinear OPF problem needs to be linearized to a LP form. It has also discussed that the OPF problem could be solved by LP methods efficiently, but the approximated LP form may not be sufficient for capturing the nonlinear behaviour of OPF problem and thus LP methods may likely provide a local optimum especially for the highly nonlinear OPF problems. Instead of linearizing the OPF problem, the nonlinear OPF could also be solved directly by using the gradient information obtained from the first order derivative of OPF problem (i.e. Nonlinear Programming). Such as [41] first applied Reduced Gradient (RG) method to solve nonlinear OPF problem in the following form

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T[\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})] \tag{2.16}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}}\ \boldsymbol{\lambda} = 0 \tag{2.17}$$

$$\nabla \boldsymbol{f} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = \frac{\partial f}{\partial \boldsymbol{u}} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{u}}\ \boldsymbol{\lambda} = 0 \tag{2.18}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = 0 \tag{2.19}$$

$$\boldsymbol{u}^{t+1} = \boldsymbol{u}^t - \boldsymbol{c}\nabla \boldsymbol{f} \tag{2.20}$$

where (2.16) is the augmented Lagrangian objective function that converts the constrained problem to unconstrained problem by adding $\boldsymbol{\lambda}^T[\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})]$, $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers, $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ is the Jacobian power flow matrix, and the gradient $\nabla \boldsymbol{f}$ measures the sensitivity (i.e. gradient) of the Lagrangian objective function with respect to the change in control variables $\boldsymbol{u}$. (2.20) shows that

the control variables are updated against the gradient of the Lagrangian objective function, while the update toward optimum usually exhibits a well-known "zig-zag" search path [41]. It has been claimed in the paper that the step length coefficient $\boldsymbol{c}$ has significant effects on the convergent performance: a small value of $\boldsymbol{c}$ could ensure the convergence but it may not efficient as more iterations are needed, whereas a large value of $\boldsymbol{c}$ may cause divergent issue, which could make the iterative process to cycle around the optimum. On the other hand, the RG method has difficulty in properly dealing with inequality constraints, as it can be seen that the Lagrangian function does not consider the inequality constraints. Although the enforcement strategy and the fuzzy penalty method were proposed in [41] to handle the limits on control and state variables, respectively, it is noticed that the poorly defined penalty factors may cause ill-condition leading to very slow convergence or even divergence problem. In [133], it first applied the Generalized Reduced Gradient (GRG) method to handle the inequality constraints in an effective way – if any violation is detected, the violated state/control variables are enforced to their limits and repartitioned as control/state variables. The repartition concept is based on LP sensitivity equations that the required changes in state variables could be derived directly from control variables, and vice versa, as shown in (2.7) and (2.8). In addition, an adaptive way, based on the terms $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$ and $\frac{\partial f}{\partial \boldsymbol{x}}$, used to calculate the step length coefficient was also proposed in [133], but it was not numerically tested in that paper. The gradient based methods, however, still cannot fully address the most concerned local optimum issues, because they can only guarantee to find a stationary point (may not be the global optimum or even local optimum for the nonconvex optimization problem) where the gradient is nearly zero, according to the fundamental theorem of calculus [134].

2.3.1.3 **Newton based method**

The RG methods using first-order derivative technique can only locates the stationary point of objective function, but it may be a saddle point or a local optimum. This has driven the development of Newton based methods, which could use the second-order derivative technique to improve the computational accuracy when solving nonlinear OPF problems. For example, [135] proposed a Newton based OPF algorithm, which also employs the Lagrangian objective function

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T[\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})] + \frac{\boldsymbol{\mu}^T}{2}[\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) - \widetilde{\boldsymbol{h}}(\boldsymbol{x}, \boldsymbol{u})]^2, \tag{2.21}$$

where the last term is called quadratic penalty function and it is excluded from (2.21) if the variables are within their limits. Note, the use of penalty function is able to force the variables close to their target values described by $\widetilde{\boldsymbol{h}}(\boldsymbol{x}, \boldsymbol{u})$. The solution procedures may be described as: first the first-order derivative of (2.21) with respect to each variable should be determined to satisfy the KKT condition (i.e. derivative or gradient is set equal to zero); second the Hessian matrix (i.e. second-order derivative of (2.21)) should be defined; then the search direction could be defined according to the information of Hessian matrix and KKT condition; finally the optimal solution could be found by

using Newton method. However, this Newton method treated the inequality constraints as equality constraints, and they were handled using the similar enforcement and penalty approaches as in GRG method. In this way, the optimization may also be oscillating and divergent if the inappropriate penalty weigh $\boldsymbol{\mu}$ and enforcement strategy are used, even though Newton based method could find more accurate solution. Moreover, the Newton based methods are conceptually able to guarantee that the converged stationary point is a truly local optimum rather than a saddle point, if the Hessian matrix is positive semidefinite in the vicinity of the optimum [135]; yet it is similar to the RG methods that the converged stationary point may not be the global optimum especially the start point is not properly defined and the optimization problem is nonconvex [136]. Furthermore, the optimum can be obtained by RG or Newton based methods only if the problem is derivable (may not always available in practice) and the matrix is invertible.

2.3.1.4 **Interior point method**

As discussed before, the IPM is proposed as a replacement of SM to solve the LP problems. Nevertheless, it has been refined by many other works in order to solve the constrained NLP problems efficiently, and it is highly competitive comparing with other deterministic methods. More specifically, the difficulty and drawback of SM, RG and Newton based methods in dealing with inequality constraints was a motivative factor of using IPMs to solve OPF problems, as introduced in [42], and one of the most successful primal-dual IPM may be expressed as follows

$$min.\ f(\boldsymbol{x}) + \mu \sum_{i=1}^{n} \log(\boldsymbol{l}_i) + \mu \sum_{i=1}^{n} \log(\boldsymbol{u}_i) \tag{2.22}$$

$$s.t.\ \boldsymbol{g}(\boldsymbol{x}) = 0 \tag{2.23}$$

$$\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u} = \overline{\boldsymbol{h}}(\boldsymbol{x}) \tag{2.24}$$

$$\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{l} = \underline{\boldsymbol{h}}(\boldsymbol{x}) \tag{2.25}$$

where (2.22) is barrier objective function which the logarithmic barrier terms are added to (2.1), $\overline{\boldsymbol{h}}(\boldsymbol{x})$ and $\underline{\boldsymbol{h}}(\boldsymbol{x})$ are respectively the vectors of upper and lower limits on the variables, $\boldsymbol{l}$ and $\boldsymbol{u}$ are the nonnegative slack variables for converting inequality constraints (2.3) to equality constraints (2.24) and (2.25), respectively. In this modified form, the objective value of (2.22) will tend to infinity if the slack variables approach zero, otherwise the barrier objective value is approximately equal to the original objective value calculated by (2.1). Therefore the use of slack variables and barrier objective function is able to avoid any variable from reaching its bound unless approaching the optimal solution (similar to the IPM used to solve LP problem introduced in previous sub-section). Apparently, the use of additional slack variables increases the size of optimization problem, as their values are needed to calculate. To solve above problem, the unconstrained Lagrangian function may be considered as follow

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{l},\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = f(\boldsymbol{x}) - \boldsymbol{y}^T\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{z}^T\check{\boldsymbol{h}} - \boldsymbol{w}^T\hat{\boldsymbol{h}} - \mu\sum_{i=1}^{n}\log(\boldsymbol{l}_i) + \mu\sum_{i=1}^{n}\log(\boldsymbol{u}_i) \qquad (2.36)$$

where $\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{w}$ are the Lagrangian multiplier vectors, $\check{\boldsymbol{h}} = \boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{l} - \underline{\boldsymbol{h}}(\boldsymbol{x})$ and $\hat{\boldsymbol{h}} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{u} - \overline{\boldsymbol{h}}(\boldsymbol{x})$. After giving the first-order derivative of Lagrangian function with respect to each variable and multiplier subject to the KKT condition, the Hessian matrix and the search direction could be defined via Newton method. Under this solution process, the IPM could be considered as a variant of Newton method. Note here the state and control variables are processed identically (i.e. all variables are considered as control variables) and their Newton step changes are calculated simultaneously. At each iteration, the obtained IPM solution could remain within a feasible region imposed by the Lagrangian multipliers and the slack variables [30], whereas the solution obtained from previously discussed methods may be infeasible and it is needed to enforce back to the feasible region again. Hence, IPMs dealing with inequality constraints in this way (i.e. using the barrier objective function and slack variables) is more effective and faster than the RG and Newton based methods using direct enforcement, repartition or penalty technique. However, the selection of μ is a challenge – a suitable μ could ensure the step length toward optimum as long as possible, improving the convergent speed, but an improper μ producing a short step length result in a low convergence rate [137]. This concern has motivated the development of improved IPM algorithms. The parameter μ may be fixed, or updated along with the iterative progress, while the later adaptive manner seems to be preferable. For example, the duality gap $\mu = \frac{\boldsymbol{l}^T\boldsymbol{z} - \boldsymbol{u}^T\boldsymbol{w}}{2}$ was proposed and proved in [138], it iteratively decreases the emphasis on barrier function allowing well-defined centre search path; [139] discussed and suggested that an iteratively decreasing weight coefficient (e.g. around 0.1 times the duality gap at each iteration) could be used to improve the convergence further. Because the updated complementary condition (i.e. duality gap) $(\boldsymbol{l_i} + \Delta\boldsymbol{l_i})(\boldsymbol{z_i} + \Delta\boldsymbol{z_i})$ and $(\boldsymbol{u_i} + \Delta\boldsymbol{u_i})(\boldsymbol{w_i} + \Delta\boldsymbol{w_i})$ may not zero at early iteration, the predictor-corrective IPM has been proposed to compute the corrector steps and the duality gap based on the affine scaling predictor steps. Although this process could improve the search direction at each iteration, it needs more computational efforts (e.g. the calculation of affine scaling steps and corrector steps) than the standard IPM, the detailed explanations may be found in [140]. There are a variety of IPMs in the literature, and they are not discussed in this thesis further. To conclude, the IPMs are more reliable than above deterministic methods when solving OPF problems, because they can well capture the nonlinearity of OPF problems, the concerned inequality constraints could be handled in an efficient way, and they present a favourable computational speed [43]. Even though IPMs are relatively less sensitive to the initial start point (i.e. central path-following rather than local search around current solution point), the optimal solution of simplified convexity Lagrangian function may not be the global optimal because the OPF problems are naturally convex with many local optima. Furthermore, IPMs also share some common weaknesses with other deterministic

methods, for example, the need of derivable objective function and constraints, the requirement of invertible Hessian matrix, and the difficulty in dealing with discrete variables.

2.3.1.5 **Convex Relaxation**

The LP, RG, Newton, IPM methods discussed previously may also be named approximation techniques, since they require that the OPF problem is approximately converted to a correspondingly solvable form. Although these methods have reached a highly mature level after few decades' development and they have been widely applied in the power industry due to their excellently computational speed. However, they cannot guarantee the global optimum and sometime may even fail to converge, especially for large-scale and high constrained OPF problems [43], [141]. To overcome these crucial concerns, convex relaxation techniques have been claimed as a promising alternative in the most recent literature surveys [33], [35], [47]. Because the recovery of global optimum of original OPF problem from the relaxed OPF form can be guaranteed whenever the relaxation is exact. In particular, the Second Order Cone Programming and Semidefinite Programming relaxations have been extensively studied. Different from the approximation techniques simplifying the OPF problem under certain valid assumptions to closely represent the power system behaviour, the relaxation techniques usually enclose the nonconvex OPF solution space in a lager relaxed convex solution space, as illustrated in Fig. 2.4 [48]. It can be seen that the convex relaxation is able to provide a certificate of solution infeasibility (i.e. an infeasible solution of relaxation means the original OPF solution is infeasible) and a lower bound to nonconvex problem (which allows ascertaining the degree of optimality of the solution), while the approximation techniques are unable to do so [33]. As long as the relaxation is tight and exact, producing zero duality gap, the optimal solution of original OPF can be retrieved from a solution of the relaxed convex problem, otherwise only a local optimum or even nonphysical meaningful solution is recovered [25], [49]–[54], [142]–[144]. However, it is worthy to mention that is not easy to contribute

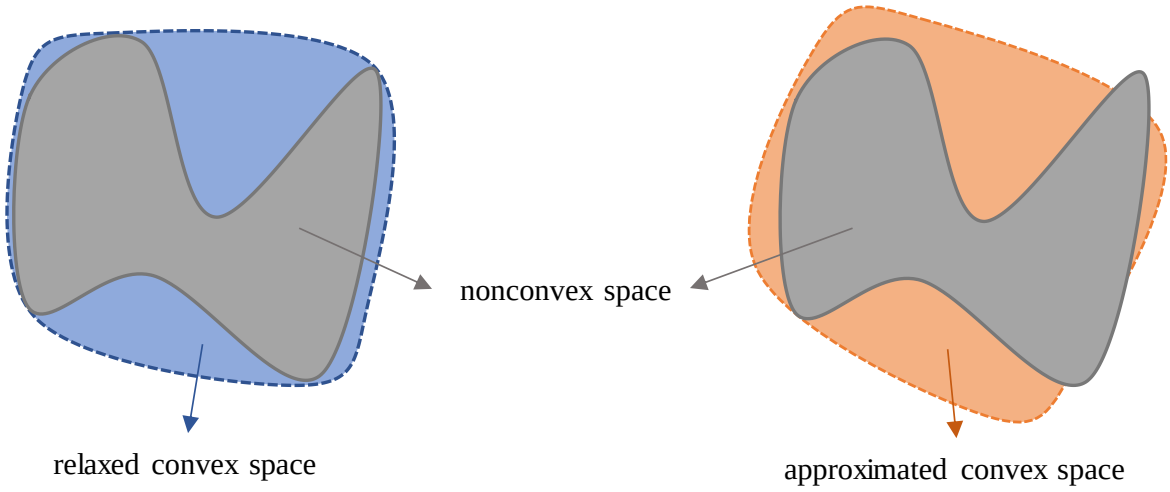


Fig. 2.4 Conceptual illustrations of relaxation and approximation

the tight and exact relaxation for a variety of OPF problems, as the relaxations work on some special OPF problems but fail to other cases.

A standard form for the SOCP is that the LP considers additional convex quadratic constraints as follows [145]

$$min.\ \boldsymbol{c}^T\boldsymbol{x} \tag{2.37}$$

$$s.\,t.\ \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.38}$$

$$\| \boldsymbol{E_i}\boldsymbol{x} + \boldsymbol{b_i} \| \leq \boldsymbol{g_i^T}\boldsymbol{x} + \boldsymbol{d_i}, \qquad i = 1 \cdots c, \tag{2.39}$$

where $\boldsymbol{E_i}$ is a matrix, $\boldsymbol{g_i}$ and $\boldsymbol{d_i}$ are vectors, $\|\cdot\|$ is the standard Euclidean norm, and c is the number of SOCP constraints (i.e. convex quadratic constraints). The SOCP based OPF was first proposed in 2006 [118], but it was only applied to the radial network. A year later, the author proposed another iterative solution procedure to extend the SCOP to the meshed network [146], yet it cannot deal with the voltage angles. In [147], an improved branch flow model was proposed in order to account the voltage angle relaxation, while it proved that the exactness of SOCP can only be guaranteed when the following conditions are satisfied: first there is no upper bounds for the loads of radial network; second the phase shifters should be installed in the meshed network. To further improve the SOCP, [148] proposed three strong SOCP relaxations for the meshed network with consideration of small voltage angle bounds; however, their exactness cannot be guaranteed and sometimes the optimality gap is large, producing infeasible solution for some OPF study cases. The inexactness issues of SOCPs have been further examined in [58], for example: the optimality gap can be as large as 21% if the loads of a simple 2-bus test system are changed to a certain level; the infeasible solutions are produced if the lower bounds on reactive powers are imposed, even though the voltage magnitudes are fixed and the voltage angle differences are very small.

Besides, SDP relaxation is another current research interest and the standard formulation may be expressed as follows [149], [150]:

$$min.\ \mathrm{tr}\ (\boldsymbol{C}\boldsymbol{X}) \tag{2.40}$$

$$s.\,t.\ \ \mathrm{tr}\ (\boldsymbol{A_i}\boldsymbol{X}) = \boldsymbol{b_i} \qquad i = 1 \cdots c, \tag{2.41}$$

$$\boldsymbol{X} \succcurlyeq 0, \tag{2.42}$$

where tr $(\cdot)$ denotes the trace operator, $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{X}$ are the specially designed square, symmetric matrices, $\boldsymbol{X} \succcurlyeq 0$ means the matrix $\boldsymbol{X}$ is a semidefinite matrix, and the matrix variable $\boldsymbol{X}$ is converted from $\boldsymbol{x}^T\boldsymbol{x}$. Due to the fact that the $\mathrm{tr}\ (\boldsymbol{A_i}\boldsymbol{X}) = \sum_i \sum_j \boldsymbol{A_{ij}}\boldsymbol{X_{ji}}$, the constraint (2.41) as well as the entire problem are considered as linear and convex [35]. In [45], [151], the OPF and Security Constrained Unit Commitment (SCUC) problems were first generalized to SDP forms, which could be solved by the proposed primal-dual IPM algorithm with superliner convergence. However, the exactness of the

SDP relaxations was not proven in that time, hence work [49] suggested an improved dual SDP model to OPF and two zero-duality-gap conditions to guarantee the exactness: first the optimal solution matrix of the dual SDP relaxation has Rank 1; second the dual SDP has an optimal solution that the positive semidefinite matrix has a zero eigenvalue of multiplicity 2. Nevertheless, the paper also claimed that it is difficult analytically study condition 1 as it can only be checked after solving the SDP relaxation, while the condition 2 may only be satisfied in the certain OPF test cases – like the resistive networks with active power loads only, the networks with no reactive load constraints, and the small resistance is necessarily added to each transform that originally has zero resistance. In addition to above necessary and sufficient conditions, there are also many other works focused on developing exact SDP relaxation for OPF problem, for example, it is proved in [152] that the exact SDP relaxation of OPF problem exist in a lossless tree network with only voltage magnitude inequality constrains, which is similar to a simplified DC network model. Moreover, an exact SDP relaxation proposed in [153] works on a weakly-cyclic network where the cycles of size 3, the line flow limits are only specified by the differences in voltage angles, and no lower bounds on the power injections. Under this concern, the exactness of this SDP relaxation has been further proven in a lossy cyclic network of size 3 where all lines have same R/X ratios and the lossless cyclic network of size 4 or n [154]. Nonetheless, the power injection models of these networks are bounded, apparently, such inequality power injection models may result in that the loads arbitrarily increase beyond the specified limits, while realistic stead-state load model should be fixed. These conditions, however, may not always be satisfied in practical power systems, because these SDP relaxations are also developed for special classes of OPF instances. Such as, [56] first demonstrated that the non-zero duality gap arise in the SDP relaxation of OPF problems when the line flow limits, the active power limit at slack bus, or the voltage magnitude constraints at PQ buses are imposed, yielding nonphysical meaningful solution. (e.g. voltage magnitude violations); [57] examined that the tighter bounds on the voltage magnitudes, the line flow, or the reactive power generation can lead the SDP relaxation failing to meet the condition 2, producing local optimum or even infeasible solution; [155] tested the SDP relaxations on a variety of standard IEEE test systems, it is found that the optimality gaps are not always being zero and for some cases the gap can be greater than 30%.

In fact, the SOCP relaxations are somehow incomparable with SDP relaxations, because all of them strongly depend on their associated exactness condition to find the global optimum. State-of-the-art convex relaxations and their exactness conditions are reviewed in [35], [48]. It should be noted that both SOCP and SDP relaxations may give same lower bound to the OPF problem if they are inexact, and thus their simulation result used to indicate their inexactness could stand for each other, as prove and studied in [58], [156]. Moreover, the computational efficiency of SDP relaxations is significantly slower than SOCP relaxations and approximations techniques [43], [59], [60], [148], [155], [157], especially in the large-scale OPF cases (e.g. bus number over 1,000), because the size of positive semidefinite matrices grow as $O(n^2)$, where n is the number of variables. Furthermore, the convex relaxations satisfying some nontrivial technique assumptions seem to be more promising

in finding global optimum than the approximation techniques satisfying KKT condition only; the convex relaxation techniques are not yet ready to use in practice, because they are not capable of solving the OPF problems that consider the practical operating conditions of interest. Comparing the approximation techniques with the relaxation techniques, the following conclusions may be drawn: the approximation techniques are inherently local solvers that critically rely on both the convexity of OPF problem and the hard to be defined "good" start point to find the global optimum, nevertheless, they are able to produce a reasonable OPF solution within an acceptable computation time, so they have been widely applied to industry application and commercial software; the convex relaxations so far are not as reliable as the approximation techniques, although they are capable of convexifying the nonconvex OPF problem, but their strictly required exactness conditions are not always satisfied in the practical power systems, therefore they may produce local optimum and even infeasible solution to OPF problems.

2.3.1.6 **Summary**

The fundamentals, strengths and weaknesses of six representative deterministic methods – including SM, RG, Newton, IPM (for LP and NLP), SOCP and SDP for solving OPF problem are introduced and discussed in this section. They could be able to gradually alleviate the difficulties in terms of computation speed, convergence, start point, and local optimum. Although over the years many efforts have been made in order to improve the deterministic based OPF algorithms, there are still some issues needed to address, because the nonlinear and nonconvex OPF problem with the discrete variable (i.e. MINLP OPF problem) is naturally very difficult to be solved. Mathematically, the deterministic methods based on the locally derivative gradient information to derive the final solution, while the discrete variables are not derivable and the appropriately start point is not easy to define, which may easily lead to the sub-optimum or local optimum. Besides, the deterministic methods are usually associated with the specified programming formulation to solve the OPF in the approximately or relaxingly convex and continuous form (e.g. LP, QP, NLP, SOCP or SDP). However, it is very difficult to fully eliminate approximation or relaxation error to ensure that the optimal solution found in such approximated OPF problem is feasible and same as in the original OPF problem. (i.e. producing local optimum or infeasible solution). Moreover, the deterministic methods are inflexible and inconvenient to be used from the programming point view, due to the fact that the Jacobian matrix (i.e. first-order derivative) and the Hessian matrix (i.e. second-order derivative) for each OPF problem has to be derived individually.

2.3.2 **Evolutionary Computation**

2.3.2.1 **Introduction**

As introduced and discussed in previous section, over the years many efforts have been made, in terms of convexification, convergence, handling inequality constraints and sensitiveness of selecting

start point, to improve the solution quality of deterministic methods when solving OPF problem; however, these issues have not been fully addressed. Generally, the deterministic methods use locally derivative gradient information of both objective function and constraints to derive optimal solution from the convexified OPF problem (e.g. approximated or relaxed) of which may not capture the accurate behaviour and the truly global optimum of original OPF problem. Consequently, these methods may still produce local optimum or even infeasible solution to the nonlinear, nonconvex and highly constrained OPF problem with discrete variables. From programming point view, the deterministic methods are inflexible and inconvenient to implement, because the Jacobian matrix (i.e. first-order derivative) or the Hessian matrix (i.e. second-order derivative) or both (depend on the selected solution technique) has to be specifically derived for each type of OPF problem [45]. Nevertheless, it has to be emphasized that they are preferable in the power industry, because they usually produce the reasonable OPF solution within the acceptable time.

In addition to the deterministic methods, the evolutionary computation (or heuristic/stochastic algorithm) has been studied extensively in the literature, because they can easily deal with some of technical issues that exist in the deterministic methods. As reviewed in [31], [120], [158], [159], the evolutionary computation may has following main advantages:

- The derivative-free evolutionary computation directly uses the objective function value to guide the search direction, not the derivative based knowledge as in the deterministic methods. Therefore, using the evolutionary computation to solve OPF problem can be considered as a "black-box" optimization process, which is adaptive to solve any type of OPF problem.
- The discrete variables and functional constraints (e.g. transient stability) needed to consider in the OPF problem can be accurately represented in the evolutionary computation through the encoding-decoding process and the additional functional analysis, respectively.
- The evolutionary computation usually starts with multiple initial solution points, enabling the achievement of multiple paths parallelly searching the optimal solutions at each iteration. This is a more reliable way to find the global optimum than the deterministic methods that use a single initial point and single path to approach the optimal solution gradually.
- Considering the optimization procedure, the solution of evolutionary methods can move over hills and across valleys in the solution space, unlike the solution of the deterministic methods may be stuck at any hill or valley which has zero or nearly zero gradient. Technically, the evolutionary computation not only locally searches around current optimum (i.e. exploitation process) but also globally searches outside locally optimal region (i.e. exploration process), allowing for roughly searching the whole solution space and locating the global optimum.

Therefore, the evolutionary based algorithms are conceptually more robust and reliable than the deterministic methods, as they are able to find the global optimum for the OPF problem of any accurately desired form if sufficient search time is given, no matter the complexity (i.e. shape of the problem or nonlinearity and nonconvexity), variables and constraints. However, the evolutionary

computation is actually a random search method, which usually requires extensive number of calls for objective function evaluation and feasibility check, dramatically increasing the computational cost, despite it is much more intelligent and efficient than the exhaustive search. This issue has mainly limited the application of evolutionary based algorithms in practice, especially in the Real-Time applications that need the OPF to be solved in the matter of minutes or seconds [8]. For example, a Teaching-Learning based evolutionary computation would take few minutes to obtain the optimal generation cost for a 118 bus test power system [70], whereas the deterministic methods only need few seconds [43]. Nonetheless, the evolutionary computation may be suitable and applicable for dealing with the time-insensitive OPF problems related to power system planning and scheduling, because they could be implemented ahead (e.g. day-ahead or yearly-ahead) the operating time [10]. Furthermore, the complexity of power system and of OPF problem increases dramatically in recent years due to the presence of new technologies (e.g. HVDC, FACTS devices, intermittent renewable generations, energy storage devices), while the computational performance of the deterministic methods will be much worse than the evolutionary computation when solving this kind of OPF problem (i.e. MINLP), as evidenced in [72], [160], [161]. For instance, a deterministic method took about 19 hours to deal with a 33-bus smart grid OPF case (49,898 variables), whereas the tested evolutionary algorithms achieved better OPF solutions in averagely 30 minutes [72]. It turns out that the evolutionary computation is very promising for the future smart gird operation.

It can be found from the literature, a number of evolutionary based algorithms - like Genetic Algorithm (GA) [61], Particle Swarm Optimization (PSO) [62], [63], Simulated Annealing [64], Artificial Bee Colony [65], Differential Evolution Algorithm [66], [67] and Hybrid Evolutionary Algorithms, have been proposed to solve different type of OPF problem efficiently. The follows brief introduce and discuss the most popular and well-known Genetic Algorithm and Particle Swarm Optimization, which have simple coding structure and are used to verify the proposed surrogate strategies in later chapter.

2.3.2.2 **Genetic Algorithm**

GA is inspired by the Darwinian theory of nature selection that the beneficial characters become more common as a reproducing population, whereas unbeneficial characters would be abandoned [158]. For the purpose of numerical optimization, GA randomly generates a population that gradually evolves toward a better target (e.g. objective function value) through several generations (i.e. iterations). As introduced in [162], the evolution process is achieved by using three main genetic operators – like selection, crossover, and mutation, which attempt to move the individuals of a population to the favourable solution region and in the process converging to a global optimum. The procedure of applying GA to solve OPF problem may be generalized as follows:

1) Initialization: this step randomly generates a population consisting of specified number of individuals (i.e. control variables like active power outputs and voltage magnitudes at PV

buses, etc.), while the crossover and mutation probabilities and the maximum number of generations are defined.

2) Evaluate the fitness of each individual: each individual in the population is used as an input of the power flow analysis to calculate the fitness.
3) Selection for reproduction: after the fitness evaluation, two individuals from the population are selected as the parents for reproduction. The selection of individuals is based on fitness that the better the fitness the higher the chance to be selected as parents.
4) Crossover: if a randomly generated number is smaller/larger than the pre-defined crossover probability, this crossover operator is performed to let the selected parents to exchange their information and generate offspring, otherwise the selected parents remain as offspring. This genetic operator is very important for the exploration (i.e. local search) that emphasises on refining the currently optimal solutions.
5) Mutation: the mutation operator is implemented if the randomly generated number at this step is smaller/larger than the pre-defined mutation probability, otherwise this step is skipped. Generally, a small number of variables in the selected offspring are mutated to different values, this process of which emphasises on exploration (i.e. global search) that allows to walk out from the current solution region found by the crossover to explore a new solution space. Hence, this mutation operator is very important for avoiding the solution being trapped at local optimum.
6) Evaluation and survival: the offspring are evaluated using power flow analysis and merged with previous population. By comparing the fitness of previous population and offspring, a certain number (same with the population size) of individuals with better fitness are selected to reproduce a new population for next generation. If the stopping criterion (e.g. maximum number of iterations and maximum allowance of computation time) is achieved, the best individual is returned as the final optimum, otherwise go back to step 3 until the stopping criterion is reached.

The specific genetic operators are not introduced here, but they could be found in [162], [163]. It should be careful that the performance of GA is largely affected by the genetic operators and the parameter settings of population size, crossover probability and mutation probability, because the operators can individually contribute to exploitation and exploration, whereas the parameter settings can be used to control the exploitation and exploration [164]. As introduced above, the crossover and mutation operators associate with the exploitation and exploration, respectively, while the probability settings generally imply the related searching power of GA (e.g. the higher the probability of crossover/mutation the higher the exploitative/explorative power, and vice versa [165]). For the selection operator, it has been characterized that higher selection pressure (e.g. larger number of candidates for selection) tends to exploitative search, in contrast, lower selection pressure prefers to explorative search [166]. Besides, a large population size would allow GA searches the solution space more thoroughly, hence increasing the explorative capability while reducing the chance of

returning local optimum; in other words, a small population size lacking exploration could cause the population being stuck at the local optimal region [162]. In addition, the setting of population size links with the computational efficacy of GA, as it directly reflects the number of objective function evaluations that needs to be performed at each generation. During the optimization process, the parameter settings can be fixed or changed accordingly. As the parametric studies presented in [167], the use of different operators and different fixed parameter settings can lead to completely different optimal solutions. In [168], an adaptive crossover and mutation probabilities was proposed and proven to be an efficient way for improving convergence rate and preventing premature convergence (i.e. converge to local optimum). Consequently, it was applied to solve reactive OPF problem with improvement in accuracy and computation time [169]. Moreover, the concept has been extended to solve OPF problem with adjusting population size during the optimization process [170]. It showed slightly better generation cost with less function evaluation requirements. Actually, the parameter setting of GA is a long-standing issue, because there are no unifying parameter settings, which can guarantee that the GA can always provide good optimal solutions to different optimization problems with good computational efficiency. [171]. Nevertheless, it is essential to find a trade-off or balance between exploitation and exploration for GA in order to achieve good solution quality and favourably computational performance [172], [173]. As the bias of exploitative search usually presents fast convergent performance, but the final solution is likely being trapped at local optimum; whereas focusing on exploration would lead to better optimum, but it may slow down the overall computational performance.

2.3.2.3 **Particle Swarm Optimization**

PSO was first introduced by Kennedy and Eberhart in 1995 [174] and it mimics the social behaviour of organisms such as animal flocking and fish schooling. During the PSO optimization process, each particle of the population can remember its personal best information (fitness and position) found by far (i.e. current generation or iteration), while the population can remember the global best information found by any particle so far. The personal best and global best information can be used to iteratively update the particles' position to find the global optimum, as described by following equations

$$\boldsymbol{v}_i^{k+1} = \omega \boldsymbol{v}_i^k + c_1 \boldsymbol{r}_1 \left(\boldsymbol{P}_{best_i} - \boldsymbol{x}_i^k \right) + c_2 \boldsymbol{r}_2 \left(\boldsymbol{G}_{best} - \boldsymbol{x}_i^k \right) \tag{2.43}$$

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \boldsymbol{v}_i^{k+1} \tag{2.44}$$

where i is the index of particle, k is the number of current generation, c_1 and c_2 are personal (or cognitive) and global (or social) learning coefficients, respectively, ω is the inertia weight and is a positive value usually less than 1, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are random number vectors with the values between 0 and 1, $\boldsymbol{G}_{best}$ is the vector of global best position, and $\boldsymbol{x}_i$, $\boldsymbol{P}_{best_i}$ and $\boldsymbol{v}_i$ are the vectors of i^{th} particle's current position, personal best position and velocity, respectively. Note that the velocity,

position and random number vectors have the same length with the number of variables. The solution procedure of PSO-OPF may be described as follows:

1) Initialization: this step first defines the maximum number of generations, the population size, the inertia weight and the learning coefficients. Then a population named current swarm consisting of specified number of particles is randomly generated, while the position of each particle is used as the input of power flow analysis and the fitness of each particle is evaluated. Next the positions and the fitness of memory swarm (i.e. consisting of personal best particles) are set equal to the current swarm. Finally, the particle that has best fitness is set to be the global best particle.

2) Update the velocity and position: according to the pre-defined parameters (i.e. ω, c_1 and c_2) and the randomly generated random number vectors (i.e. $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$), the equations (2.43) and (2.44) can be computed to obtain the velocity of each particle and update the position of each particle.

3) Evaluate the fitness and update the personal and global best information: the fitness of each updated position is evaluated via power flow analysis. If the fitness of the updated particle is better than the personal or global best fitness, the personal or global best particle will be replaced by the updated particle, otherwise the personal or global particle will remain being unchanged.

4) Check the stopping criterion: If the maximum number of generations (or other criterion) is reached, the best solution recorded during the optimization is returned as final optimum, otherwise go back to step 2.

Similar as in GA, the performance of PSO is also critically affected by the parameter settings that are associated with the exploitation and exploration. The velocity in PSO is a significant element to balance the exploitation and exploration of PSO process – the faster the velocity the more emphasis on exploration, whereas the slower the velocity the more emphasis on exploitation. While this conceptually velocity depends highly on the value of inertia weight, personal and global learning coefficients. The inertia weight parameter was first introduced to PSO in 1998 [175], which advocates that inertia weight should be set to a small value for exploitative search and a large value for explorative search. It also suggested that the inertia weight should be linearly decreasing along with the optimization process – starting with a value slightly greater than 1 and decreasing to a value less than 1 – to maintain the convergent performance. In the very first version of PSO [174], a recommended setting of personal and global learning coefficients is 2, since the random vectors ($\boldsymbol{r}_1$ and $\boldsymbol{r}_2$) multiplying by 2 could give the weight vectors with statistical mean of 1 to cognitive (i.e. $\boldsymbol{P}_{best_i} - \boldsymbol{x}_i^k$) and social (i.e. $\boldsymbol{G}_{best} - \boldsymbol{x}_i^k$) parts while achieving good convergence. In [176], Kennedy proposed and proven that the constriction coefficients (i.e. $\omega \approx 0.72984, c_1 = c_2 = 2.05$) and $c_1 + c_2 = 4.1$ may be necessary to insure convergence of PSO. It also found that the summation of c_1 and c_2 smaller than 4 may not guarantee the convergent performance and larger than 4 would improve

the convergent performance, while it is possible to adjust the values of c_1 and c_2 to favour the exploitation (e.g. increasing c_2 while decreasing c_1) or the exploration (e.g. decreasing c_2 while increasing c_1). Based on these well-developed knowledge, [177] proposed an efficient strategy to adaptively adjust the values of ω, c_1 and c_2 to balance the exploration and exploitation, while successfully improving the solution quality and the computational efficiency of PSO when solving numerical optimization problems. [178] tested 15 up-to-date inertia weigh strategies using 5 benchmark functions, while it concluded that the "Chaotic Inertia Weight" is the best strategy for better solution quality and "Random Inertia Weight" strategy is best for better efficiency. This conclusion, however, may not be reliable as it only tested on the small-scale (10 variables) problems, the inertia weight strategy should be carefully selected for OPF problems.

In addition to the parameter settings, the PSO swarm topology also plays a very important role in balancing the exploitation and exploration. In the early development of PSO [179], there are actually two proposed PSO versions, which are called global (i.e. the one introduced above) and local version that come up with "gbest" topology (i.e. as a strong meshed network where every particle is connected to every other particle) and "lbest" topology (i.e. a ring network where each particle is connected to its immediately adjacent neighbours, e.g. neighbourhood = 2), respectively. Instead of moving toward the personal best and the current global best (i.e. the best fitness in the entire swarm), the particles in the local version move toward the points specified by the personal best and the local best (i.e. the best fitness in the neighbourhood). It has been pointed out that the global version emphasises on exploitation and converges fast to the current best position of which is likely a local optimum (i.e. premature convergence); inversely, the local version emphasises on exploitation and converges more slowly, because it explores the search space more thoroughly to avoid premature convergence [180], [181]. In order further analyse the impact of swarm topology on the performance of PSO, a number of different topologies – such as wheels, pyramid and random graph (based on the "small-world" concept first proposed in [182]) – have been tested with varying results [183], [184]. Moreover, it is argued in [185] that each particle is not simply affected by its personal best and the best among its neighbours, rather it should be fully informed from all of its neighbours. However, the movement of swarm and the result may be confusion in the presence of many influences, resulting in slow convergence rate [186]. Based on these fundamentals of PSO, recent works [187], [188] have indicated that the size of neighbourhood is an efficient parameter for tuning the exploitative and explorative search power. They have also augured that a large size of neighbourhood has relatively strong exploitation capability as it is likely to select the current global best as the neighbour best (i.e. similar to the global version); in contrast, a small size of neighbourhood tends to choose the adjacent or outlying neighbour to guide the search (i.e. similar to the local version) and therefore has relatively strong exploration strength. In order to improve the swarm diversity and avoid the premature convergence, recent works also proposed the adaptive strategies based on the fitness values and the distances between particles to allocate different tasks (e.g. exploration and exploration) for the different particles or swarms (i.e. multi-swarm PSO) of which can search more thoroughly in the

solution space [189]–[191]. It is found that the solution quality (for numerical optimization problems) could be improved by these proposed strategies, but the number of necessary function evaluations is increased as well as the computational cost. Furthermore, due to the standard PSO has no mutation operator as in GA to "jump-out" the local optimum, many variants of PSO have proposed to employ the additional mutation operator for improving exploration capability of PSO and finding the global optimum, such as in [187]–[191]. The mutation strategy, however, need to be carefully used and designed to avoid increasing computational cost. To conclude, the previously introduced GA and PSO (or other evolutionary based algorithms found in the literature) may not be comparable, since their computational performance are largely influenced by their parameter settings. The fine-tuned parameters (not easy to be defined) lead the algorithm to achieve a balance between the exploitation and the exploration while avoiding premature convergence and saving computation time, whereas the poorly tuned parameters may adversely affect the performance of the algorithm. Many other evolutionary based algorithms can be found in literature and they are not discussed further in this thesis. Although the evolutionary computation is more promising for finding the global optimum than the deterministic methods, the impractically computational time must be managed or alleviated to an acceptable level when it is applied to industry.

2.3.2.4 **Summary**

In this section, two popular evolutionary based algorithms (i.e. GA and PSO) are introduced and discussed, in terms of theoretical background, pros and cons. It is found that the derivative-free evolutionary computation can well-capture the behaviour of power system, solve the OPF problem in the most accurate MINLP form without losing any system information, is easily to be implemented to solve any type of OPF problems, and is much more reliable in finding global optimum than the deterministic methods due to the exploration capability. However, the computational efficiency of evolutionary computation has limited its application in power system optimization. It is found in the literature that the computational performance of evolutionary based algorithms can be greatly improved by tuning the related parameters and balancing the exploitation and exploration, nevertheless, this is not easy to achieve due to the stochastic nature of these algorithms. To improve the computational efficiency of evolutionary computation when solving the OPF problems, this work uses and develops a surrogate based technique to reduce the total number of required function evaluations, instead of finding an approach to tune the algorithms' parameters.

2.3.3 **Other Techniques**

Rather than implementing the deterministic method or the evolutionary computation alone, the hybrid method may be a more efficient way for OPF applications [31], [158]. By using the hybrid method, technically, the deterministic method could improve the exploitation capability and the computational efficiency of evolutionary computation, whereas evolutionary computation could use to deal with the discrete variables and improve the exploration capability of deterministic method. In

other words, the hybrid method is a two-stage approach – the evolutionary computation method is used to find a solution within the globally optimal solution region for which the deterministic method can use it as an initial start point and fast convergent to a precisely final global optimum, such as the algorithms proposed in [192]–[194]. Although their results have proven that the hybrid method would outperform the algorithm implemented alone in some tested cases, the computation time is not largely improved if first stage takes many objective function evaluations to find out the initial start point needed in second stage for the determinist methods. For example, the fitness value and the convergence rate of hybrid PSO and linear IPM is only slightly better than the PSO performed alone [194]. Moreover, the obtained solution may be still a sub-optimal solution, because the deterministic method in second stage can only find the accurate optimum for continuous variables. For the evolutionary based algorithms, their parallelism makes the parallel-computing technique as a reliable and feasible way to improve their computational efficiency. However, it usually has more strict requirements on both hardware and programming. Recently, [195] proposed that using GA to solve the OPF problem in an approximated LP form. In this approach, only one individual is evaluated using power flow calculation at each iteration, while others are approximated via Jacobian power flow matrix and sensitives matrix with respect to control variables. The number of power flow calculations are substantially reduced as well as computation time; however, if the changes become large, the approximation will lose accuracy leading to sub-optimal solution and slow convergence rate. In fact, this method is conceptually similar to surrogate assisted evolutionary computation [73], which uses the approximated modelling method to approximate the objective function value instead of using expensively real objective function to reduce the total number of objective function evaluations (i.e. computational time). Moreover, the potential surrogate modelling methods and the surrogate assisted evolutionary computation, which have not been applied to power system optimization before, are discussed and developed in following chapters.

2.3.4 **OPF Preferences**

In practice, the OPF is an essential tool for power system operation and control and it must be solved many times a day, as often as every 5-15 minutes (i.e. Real-Time operation control) [8]. Due to the desirable computational performance of LP based OPF problems and solution techniques, the deterministic methods (i.e. approximation techniques) using linearized network model are the current industrial OPF preference in daily electricity market clearing and power system operations [47], even though they are considered as a trade-off between solution quality and computational efficiency. As for the full AC power flow based OPF, it is mainly used to deal with the optimal controls of reactive power and voltage magnitude due to the lack of efficient and robust algorithm [196]. On the other hand, the increasing penetration of FACTS, HVDC technologies, intermittently renewable generations, electric vehicles and energy storage devices adds more complexity to power system operation, therefore, more rigorous operation scheme may be needed so that more complex OPF is required to be solved more frequently (e.g. minute to minute) [18], [20]. The aforementioned OPF

techniques were studied to solve static power system operation and control problems (they can be extended to dynamic OPF), but there are also numbers of oriented works that deal with Real-Time OPF (RTOPF). For example, the NLP-OPF was performed every 10 min while giving Jacobian matrix and sensitivity matrix to estimate the following minute to minute optimal control [197]; [17] proposed a Newton based method to approximately track the optimal operations following time-varying (every 6 sec). These methods assumed the system with small changes (e.g. generations & loads) in small timescale, but the given results may not be accurate if the system has large changes. Moreover, such Real-Time operation framework may not be feasible, because they ignored the communication delay and assumed that the control action can respond immediately. A more realistic RTOPF framework was proposed in [198], it suggested a NLP solver to solve OPF every 5 minute, while the delay was assumed to be 90 seconds and the control actions were assumed to be completed within 60 seconds. So far, most RTOPF used the traditional deterministic methods because of their computational efficiency that can obtain reasonable solution very fast. However, the inaccurate solutions (e.g. local optimal) obtained from the deterministic method may unnecessarily cost billions of dollars per year and result in waste of energy and environmental pollution [8], [19]. In contrast, the reason of not applying evolutionary based algorithms to RTOPF is that they have inefficient computational performance, requiring much longer computation time, especially for large-scale power system; although they can provide a better operational solution. Nonetheless, attempts have been made in using evolutionary computation to solve RTOPF – such as [199] used GA to solve the energy management problem of a microgrid at every 15-min period, [200] proposed a two-stage optimization strategy that used GA to solve day-ahead OPF and NLP to solve RT-OPF, and [201] proposed a Modified Honey Bee Mating Optimization to solve the hourly dynamic OPF problem. However, it is noticed that the evolutionary based algorithms were performed on a slower timescale (e.g. day-ahead or hourly-ahead) than the deterministic methods (e.g. real-time) and tested on the small-scale power systems. It may turn out that the current evolutionary computation is applicable or the computation efficiency of evolutionary computation is acceptable for the day-ahead and hourly-ahead OPF applications that are relativity less sensitive to the computation time, however, further efforts must be made to advance the evolutionary computation to RTOPF applications. Furthermore, it is worthy to mention again that current deterministic methods are not applicable for the future smart grid OPF problems that has extensive number of control variables, particularly the binary and discrete variables, because they would require extensive computation time and massive computer memory; whereas the evolutionary computation is a more promising approach [72], [160], [161].

2.4 Summary

This chapter introduces and discusses the OPF, in terms of formulations, solution techniques and the industry preferences. Overall, one of main bottleneck of OPF throughout the literature is that the current OPF solution techniques cannot address the computation accuracy and speed issues

simultaneously. All the current OPF techniques have significant advantages in special area and disadvantages in others. Although the determinist methods using linearized network model are preferable in the industry due to their excellent computation speed, they somehow sacrifice the solution quality and cannot properly solve the full ACOPF. While a full ACOPF can generates better electricity market signal and model the exact system state to secure the system reliability and efficiency. Furthermore, evidences have shown that current deterministic methods may no longer applicable for future smart grid operation and the power system is needed to be adjusted more frequently to maintain the system reliability and efficiency. Despite the evolutionary computation could be used to time-insensitive and smart grid OPF applications, it may still not applicable for RTOPF, which requires the reliable solution that must be delivered within short computation time. It is found the parameters of evolution computation significantly influence the computational performance, while they are very difficult to find a unifying setting for different OPF problems. Under these concerns, this work is highly motivated to develop a technique without relying on the tuning the parameters to improve the computational performance of evolutionary computation when solving OPF problems.

Chapter 3 Surrogate Modelling and Optimization

3.1 Introduction

The high fidelity scientific and engineering simulations (e.g. Finite Elements Analysis-Design, Electromagnetic Design, Aerodynamic Design), requiring enormous computational cost, are usually impractical for the purpose of optimization. This concern has driven the development of computational efficiency approximation model (also known as surrogate model or metamodel) to replace the expensive simulation model during the optimization process. By sampling a set of samples from simulations and fitting a favourable surrogate model to these sampled input-output data, relationships between inputs and outputs could be modelled to locate the optimal solution. However, there are no clear conclusions to answer which surrogate technique is the best for optimization purpose, because the modelling performance is problem dependent and many criterions must be considered. The most obvious factors may be the modelling accuracy and the selected sample size and computational complexity, as they are the most important factors relating to the solution quality and the computational efficiency. In addition to these factors, this chapter analyses and discusses their capability to balance the exploitation and the exploration, which is not insignificant for surrogate based optimization methods. Three representative and promising surrogate modelling techniques, including Polynomial Regression (PR), Radial Basis Functions (RBF) and Kriging, are introduced and tested in following sections. Moreover, the Kriging based optimization equipped with exploitation and exploration capabilities is mainly explained and discussed. Furthermore, without loss of generality, the optimization problems mentioned in this work refers to minimization problem.

3.2 Polynomial Regression

Polynomial Regression is a well-known method which uses polynomial function with specified degree to construct a simple and fast approximation of simulation model. The generalized polynomial model may be expressed as follow [202]

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{p=0}^{m} \boldsymbol{x}^p \boldsymbol{\beta}_p \tag{3.1}$$

$$SSR = \sum_{i=1}^{n} [\hat{\boldsymbol{y}}(\boldsymbol{x}_i) - \boldsymbol{y}(\boldsymbol{x}_i)]^2, \tag{3.2}$$

where $\boldsymbol{y}(\boldsymbol{x})$ denotes the vector of sampled objective function values, $\boldsymbol{x}$ represents the sampled point matrix consisting of n number of d-dimensional vectors, p is the polynomial degree, m refers to

specified maximum polynomial degree, $\boldsymbol{\beta}_p$ is the vector of p^{th} polynomial coefficients, $\boldsymbol{y}(\boldsymbol{x}_i)$ and $\hat{\boldsymbol{y}}(\boldsymbol{x}_i)$ is the sampled and the predicted values at $\boldsymbol{x}_i$, respectively, and SSR is the Sum of Squared Residuals. The values of $\boldsymbol{\beta}$ could be calculated by using Ordinary Least Square (OLS) estimator to minimize (3.2) and the corresponding $\boldsymbol{\beta} = (\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T\boldsymbol{y}$ minimizes SSR is known as OLS solution [203].

When building a PR model, it is important to identify the sample size and the suitable degree of polynomial function. For any surrogate method, the number of samples is the one of most direct factors influencing the accuracy of the model, but the large number of samples may be too difficult to calculate or obtain in practice. On the other hand, the PR model's degree of polynomial also influences its capability of capturing the behaviour of original function. For example, a linear or second-order PR model may be adequate for modelling convex problems. However, the fidelity of linear or second-order PR model may dramatically decrease when applying to model a highly nonlinear problem, which is known as the main drawback of PR. In order to illustrate the impacts of polynomial degree and sample size on building PR model, eight PR models were tested based on the highly nonlinear Schwefel Function [204]. The formulation is expressed as

$$\boldsymbol{y} = 418.9829 - \boldsymbol{x} * sin\left(\sqrt{|\boldsymbol{x}|}\right) \qquad \boldsymbol{x} \in [-500, 500], \tag{3.3}$$

Two sets of samples with step sizes 100 and 10 (i.e. 11 and 101 samples) were used to build the 2nd, 7th, 11th and 19th PR models, as plotted in Fig. 3.1 (a-d) and Fig. 3.1 (e-h), respectively. Note that the true behaviour of Schwefel function (i.e. blue curve) is simulated using 1001 samples with step size 1 (same for the following tests). It can be observed from Fig. 3.1 (b-d) that the PR models (i.e. 7th, 11th and 19th order PRs) are overfitting (orange curve), especially at both ends, along with the increase of polynomial order when the samples are limited to a small number (i.e. 11 samples); whereas a much larger number of samples shows that higher the polynomial order the more accurate the model tends to be (see Fig. 3.1 (e-h)). This means that the use of higher-order polynomials with more samples could significantly improve the accuracy of the PR model. However, it may be impossible to obtain sufficient samples in practice and it is too difficult to calculate all polynomial coefficients for the high-dimension problems because the inversion of matrix $\boldsymbol{x}'\boldsymbol{x}$ is required [77]. Considering that the OPF problems are large scale, highly nonlinear, and nonconvex formulation, the PR method requiring a large number of samples are not considered for this work. Finally, the relevant Matlab codes may be found in Appendix A1.

3.3 Radial Basis Function

Radial basis function has been developed to deal with the multivariate interpolation problem [205]. It has been proven that RBF provides good fits to large dimensional and different degree of

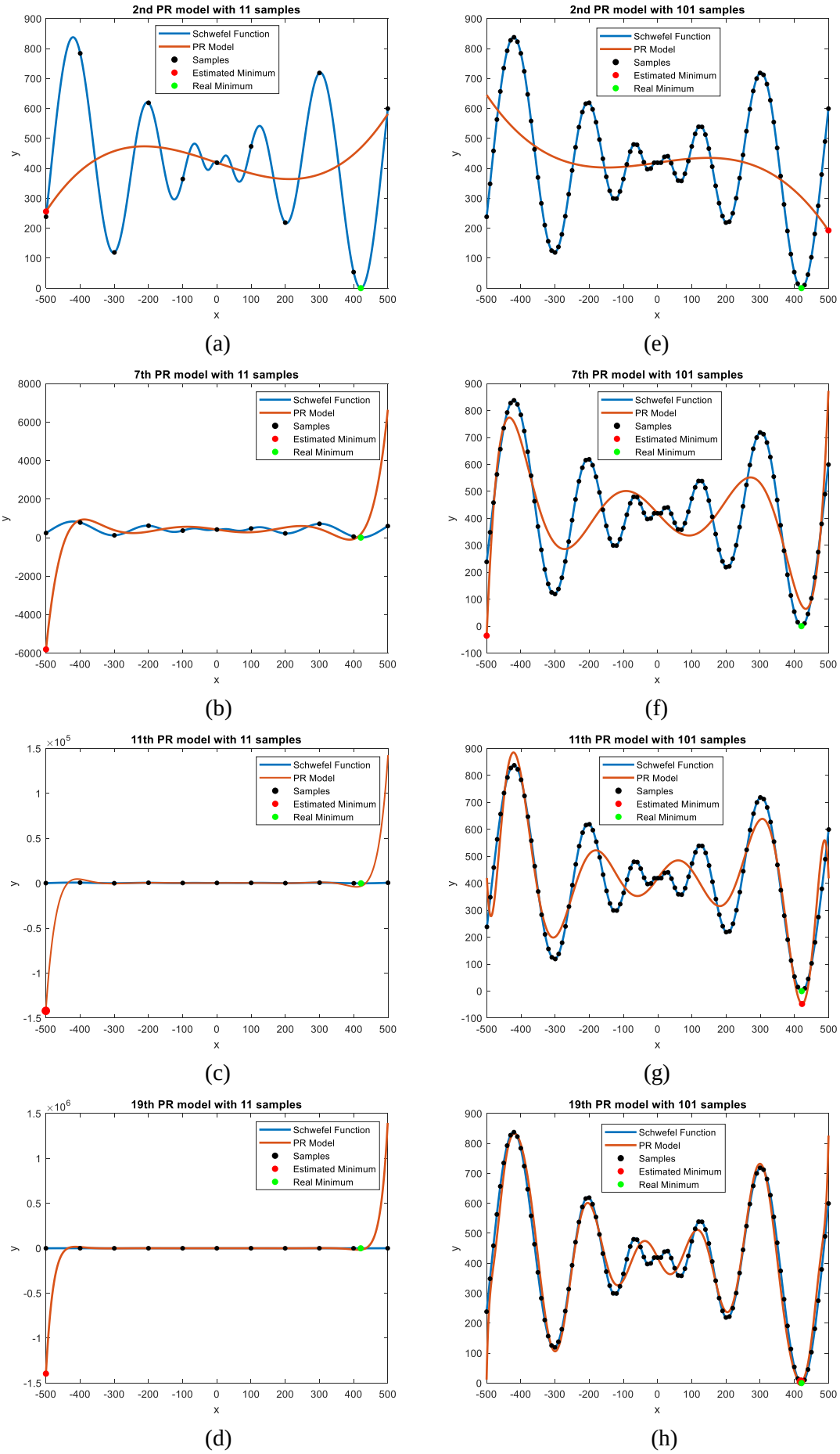


Fig. 3.1 Schwefel Function and different PR models

nonlinearity problems, using small size of training samples [206], [207]. Unlike the PR model using specified degree of polynomial to build a model, this method, namely, uses linear combination of radial distance based symmetric functions to build an approximation model as

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{i=0}^{n} \boldsymbol{\lambda}_i \varphi(\|\boldsymbol{x} - \boldsymbol{s}_i\|), \tag{3.4}$$

where n is the sample size, the coefficient $\boldsymbol{\lambda}_i$ may also be estimated using OLS, the pairwise Euclidean distance $\|\boldsymbol{x} - \boldsymbol{s}_i\|$ (denoting by $\boldsymbol{r}$) between predicting point $\boldsymbol{x}$ and sample $\boldsymbol{s}_i$ is usually used, and the φ is basis function which may have following choices [207]:

$$\begin{aligned}
&\varphi(\boldsymbol{r}) = r && \text{(linear)} \\
&\varphi(\boldsymbol{r}) = r^3 && \text{(cubic)} \\
&\varphi(\boldsymbol{r}) = r^2 \log(r) && \text{(thin plate spline)} \\
&\varphi(\boldsymbol{r}) = \sqrt{r^2 + \gamma} && \text{(multiquadric)} \\
&\varphi(\boldsymbol{r}) = e^{-\gamma r} && \text{(Gaussian)}
\end{aligned} \tag{3.5}$$

where γ is a user defined positive constant. Note the Gaussian basis function is the most common choice due to its robust performance on many nonlinear problems [206]. Furthermore, this Gaussian form is also widely used as the primary correlation function in Kriging, which is discussed in detail in the next section.

In order to simply illustrate the performance, RBF surrogate models using different basis functions were tested on Schwefel function, but the step sizes were set to 100 and 20 (i.e. 11 and 51 samples) since the RBF could build an accurate model using less samples as mentioned before. Fig. 3.2 shows the modelled RBFs and Schwefel function. The related codes of this test can be found in appendix A2. In this case, the modelled behaviours of cubic, thin plate spline and multiquadric RBF models are comparable and not distinctly improved by increased sample size, although their estimated minimums (red dot) are getting closer to the real optimal solution (green dot) when the sample size is increased. Moreover, the Gaussian RBF clearly outperforms others, and the model is greatly improved by the increased sample size. One possible reason of making Gaussian function more successful among other basis functions may be the behaviour of highly nonlinear problems from one sample to another could not be simply and fully presented by linear, cubic, spline or multiquadric; whereas the Gaussian function could provide a statistical sense to measure the relationship between samples and thus quantifies the problem behaviour. Comparing RBF model with PR suggests that RBF is more reliable and preferable in modelling complex nonlinear problems, as RBF required much less samples. However, RBF and PR models may be difficult in balancing the exploitation and the exploration when applying to solve global optimization problems, as all surrogate methods depend on the estimated optimum may be suffering from “Curse of Uncertainty”

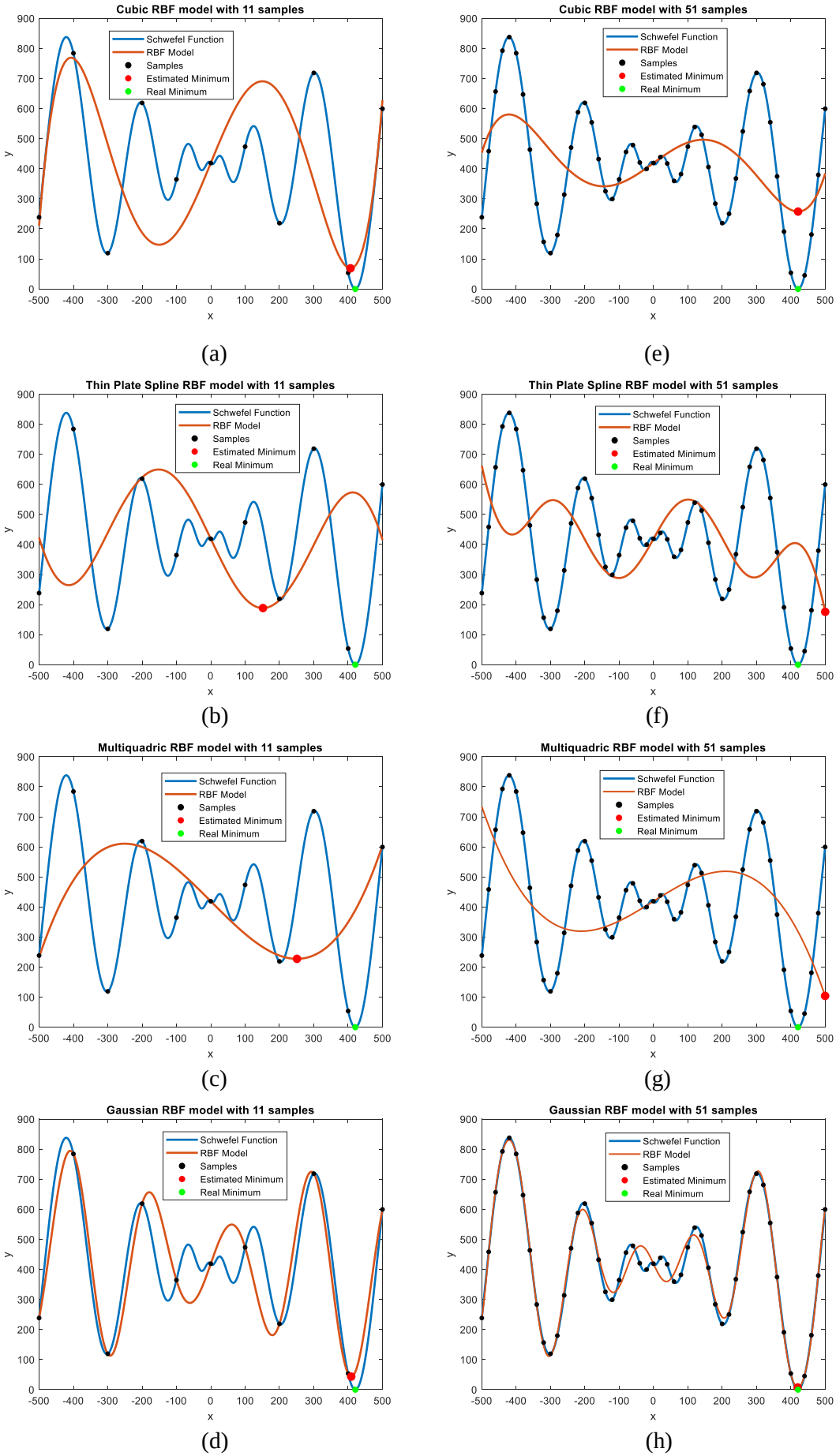


Fig. 3.2 Schwefel Function and different RBF models

(i.e. approximation error leads to local optimum) [75]. For this reason, the RBF are also not considered for this work. In next section, the selected Kriging providing exploration and exploitation capability are discussed, while an example is given to explain the weakness of using estimated optimum for solving optimization problem.

3.4 Kriging

3.4.1 Gaussian Process

The Kriging (also known as Gaussian Process) is a statistical-based surrogate method, which was originally aimed to use the samples obtained from a small set of boreholes to predict the geographical distribution of gold based at unexplored locations. The conceptual difference between the previously introduced methods (e.g. PR and RBF) and the Kriging is that, the standard response surface methods deterministically describe what the modelled function is while Kriging method describes how the modelled function is likely to vary within some range (i.e. Gaussian Process). The relevant Matlab codes of Kriging Predictor can be found in the Appendix A3. Actually, it is a simplified and efficient (i.e. programming structure and computing efficiency) version of the well-developed Kriging DACE Matlab toolbox [208]. In this section, the fundamentals of Kriging are introduced, and the modelling performance of Kriging is tested.

The Kriging may be expressed as [74]

$$\hat{\boldsymbol{y}}(\boldsymbol{x}^*) = \boldsymbol{x}^*\mu + \boldsymbol{\epsilon}(\boldsymbol{x}^*), \tag{3.6}$$

where $\boldsymbol{x}^*$ is the vector of new points to be estimated, $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ is the estimated value at $\boldsymbol{x}^*$, μ is the unknown mean of the Gaussian Process (GP), and $\boldsymbol{\epsilon}(\boldsymbol{x}^*)$ is assumed to be the normally distributed errors with mean value of zero and variance σ^2. Intuitively, this equation could be interpreted as – the predicted value at $\boldsymbol{x}^*$ is assumed to be random variable which is uncertain and likely to vary within some range $\mu \pm \sigma^2$. Under this assumption, there are so many circumstances could be used to fit the observed data, but the Kriging would suggest a most confident model that the behaviour is most confidently consistent with the observed data.

Comparing (3.3) and (3.6) shows that the Kriging has similar formulation with PR, but an additional stochastic term $\boldsymbol{\epsilon}$ is introduced to improve the accuracy of prediction. The use of OLS to estimate the value of μ may result in bias solution, as it assumes the errors are uncorrelated and have equal variance [209]. Alternatively it is suggested that the Generalized Least Squares (GLS), which takes the correlated errors and the unequal variance into account, could be used to ensure the Kriging is a Best Linear Unbiased Predictor (BLUP) [210]. The main idea behind GLS is that the OLS model

is transformed into an equivalent model, which weights the observations and the variance, to satisfy the constant variance. The transformed GLS model may be expressed as

$$\boldsymbol{W}^{-1}\boldsymbol{y} = \boldsymbol{W}^{-1}\boldsymbol{x}\mu + \boldsymbol{W}^{-1}\boldsymbol{\epsilon} \qquad E[\boldsymbol{\epsilon}] = \boldsymbol{0} \quad \mathrm{Var}[\boldsymbol{\epsilon}] = \sigma^2\boldsymbol{R}, \tag{3.7}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the vector of observations, $\boldsymbol{R}$ is recognized as the symmetric weight or correlation matrix, and $\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{R}$. Now the variance of transformed/weighted errors satisfying the constant variance is derived as

$$\mathrm{Var}[\boldsymbol{W}^{-1}\boldsymbol{\epsilon}] = \sigma^2\boldsymbol{W}^{-1}\boldsymbol{R}\boldsymbol{W}^{-1} = \sigma^2\boldsymbol{I}. \tag{3.8}$$

Therefore the least square equation used to minimize SSR can be re-defined as

$$SSR = (\boldsymbol{y} - \boldsymbol{x}\mu)\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{x}\mu)^T, \tag{3.9}$$

As can be seen that the added $\boldsymbol{R}^{-1}$ in (3.9) correlates the observed errors as well as the observations. The formulations of mean value and unequal variance of the Kriging model are introduced and derived in later section.

To correlate the errors, the basis function used in RBF could also be used as the correlation function for Kriging, but the Gaussian form is widely used [74], [211] as

$$\boldsymbol{corr}[\boldsymbol{\epsilon}(\boldsymbol{x}_i), \boldsymbol{\epsilon}(\boldsymbol{x}_k)] = \prod_{d=1}^{N} exp\,(-\widehat{\boldsymbol{\theta}}^d|\boldsymbol{x}_{id} - \boldsymbol{x}_{kd}|^{\boldsymbol{p}^d}) \qquad \boldsymbol{corr} \in [0,1], \tag{3.10}$$

where d is the d^{th} dimension of variables $\boldsymbol{x}$, N is the total number of dimensions. Note that the parameter $\widehat{\boldsymbol{\theta}}^d$ adjusts how fast the correlation drops away from one point to another in d^{th} dimension, mathematically, it could be considered as a weight factor of the correlation between any two points. Moreover, the value of $\boldsymbol{p}^d$ in general may be taken between 0 and 2 (also other values) – a value of 2 (i.e. Euclidean distance) helps to smooth the line between two points as well as the whole model, whereas a value of near 1 could produce a less smooth model – as discussed in [212]. For convenience, $\boldsymbol{corr}$ is denoted as $\boldsymbol{R}$, that is, a $n * n$ (n is the number of samples) correlation matrix which – in terms of (i, k) elements (for $i, k = 1, \cdots, n$ and the diagonal terms are 1) – provides the calculated pairwise correlations between pairs of samples. Such distance-based correlation could be interpreted as - if the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ tends to zero, their errors are highly correlated and thus the correlation is 1; whereas if the between $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ tends to infinity, their errors are not correlated and thus the correlation is zero. Moreover, for example if the predicting point $\boldsymbol{x}^*$ is very close to the observed point $\boldsymbol{x}_i$, the predicting value of $\widehat{\boldsymbol{y}}(\boldsymbol{x}^*)$ is approximated as the observed value $\boldsymbol{y}_i$ plus the correlated error between $\boldsymbol{x}^*$ and $\boldsymbol{x}_i$. Therefore, it is significant to determinate the correlations for fitting a model to observed data.

3.4.2 Maximum Likelihood Estimation

As introduced before, the deviations of Gaussian Process (GP) are quantified by the unknown parameters μ, σ^2 and $\boldsymbol{\theta}_d$. These parameters reflect how a modelled Kriging behaved, and these could be found by maximizing the following likelihood function of the observed data

$$\mathcal{L}(\boldsymbol{y}|\mu,\sigma^2) = \left[(2\pi)^{-\frac{n}{2}}(\sigma^2)^{-\frac{n}{2}}|\boldsymbol{R}|^{-\frac{1}{2}}\right] exp\left[-\frac{(\boldsymbol{y}-\boldsymbol{x}\mu)^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{x}\mu)}{2\sigma^2}\right], \tag{3.11}$$

$$\mathcal{L}(\boldsymbol{y}|\mu,\sigma^2) = \left[-\frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|\boldsymbol{R}|)\right] - (\left[\frac{(\boldsymbol{y}-\boldsymbol{x}\mu)^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{x}\mu)}{2\sigma^2}\right], \tag{3.12}$$

where (3.11) defines that the observed data is normally distributed with respect to the introduced correlation matrix $\boldsymbol{R}$, and (3.12) is the logarithmic formulation of (3.11) ignoring constant term $(2\pi)^{-\frac{n}{2}}$ for computational convenience. Choosing these parameters to maximize the log-likelihood function means that the modelled function behaviour is most confidently consistent with the observed data. To achieve the maximum likelihood of observed data, the optimal $\hat{\mu}$ and $\hat{\sigma}^2$ may be calculated by taking partial differentiation of (3.12) with respect to each parameters subject to KKT necessary condition as follows

$$\hat{\mu} = \frac{\boldsymbol{x}^T\boldsymbol{R}^{-1}\boldsymbol{y}}{\boldsymbol{x}^T\boldsymbol{R}^{-1}\boldsymbol{x}} \tag{3.13}$$

$$\hat{\sigma}^2 = \frac{(\boldsymbol{y}-\boldsymbol{x}\hat{\mu})^T\boldsymbol{R}^{-1}(\boldsymbol{y}-\boldsymbol{x}\hat{\mu})}{n}, \tag{3.14}$$

where (3.13) is in fact a GLS solution with respect to μ, the obtained optimal mean ($\hat{\mu}$) and variance ($\hat{\sigma}^2$) are highly correlated by the correlation matrix $\boldsymbol{R}$. Substituting (3.13) and (3.14) into (3.12), a concentrated log-likelihood function is then derived for estimating optimal $\widehat{\boldsymbol{\theta}}^d$ as

$$-\frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|\boldsymbol{R}|), \tag{3.15}$$

where $|\boldsymbol{R}|$ is the determinant of correlation matrix. So far, an optimal surrogate model may be constructed with the estimated optimal parameters ($\hat{\mu}$, $\hat{\sigma}^2$, $\widehat{\boldsymbol{\theta}}^d$) to describe the best statistical behaviour of observed data. In order to estimate the value $\widehat{\boldsymbol{y}}(\boldsymbol{x}^*)$ at the new point $\boldsymbol{x}^*$, the predictions $\widehat{\boldsymbol{y}}(\boldsymbol{x}^*)$ and $\boldsymbol{x}^*$ could be added to the observation data set to form an augmented data set as

$$\boldsymbol{X} = [\boldsymbol{x}, \boldsymbol{x}^*)]^T \tag{3.16}$$

$$\boldsymbol{Y}(\boldsymbol{x}, \boldsymbol{x}^*) = [\boldsymbol{y}(\boldsymbol{x}), \widehat{\boldsymbol{y}}(\boldsymbol{x}^*)]^T, \tag{3.17}$$

Hence, the correlation matrix of the augmented data is given as

$$\widetilde{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{r} \\ \boldsymbol{r}^T & \mathbf{1} \end{bmatrix}, \tag{3.18}$$

where $\mathbf{1}$ is $n * 1$ vector of ones, $\boldsymbol{r} = \boldsymbol{corr}[\boldsymbol{\epsilon}(\boldsymbol{x}^*), \boldsymbol{\epsilon}(\boldsymbol{x}_i)]$ is a $n * m$ (m is the number of points to be estimated) correlation matrix, containing calculated pairwise distance between observed points and estimating points, similar to (3.10). Referring to log-likelihood function (3.12), as only the last term depends on both augmented data set (i.e. $\boldsymbol{Y}$ and $\widetilde{\boldsymbol{R}}$), the augmented log-likelihood function becomes

$$\tilde{\mathcal{L}}(\boldsymbol{y}|\hat{\mu}, \hat{\sigma}^2, \hat{\boldsymbol{y}}) = -\frac{(\boldsymbol{Y} - \boldsymbol{X}\hat{\mu})^T \widetilde{\boldsymbol{R}}^{-1} (\boldsymbol{Y} - \boldsymbol{X}\hat{\mu})}{2\hat{\sigma}^2}, \tag{3.19}$$

To find the optimal estimation of $\hat{\boldsymbol{y}}$ while maximizing the augmented log-likelihood function, above function is expressed as [211]

$$\tilde{\mathcal{L}}(\boldsymbol{y}|\hat{\mu}, \hat{\sigma}^2, \hat{\boldsymbol{y}}) = \left[\frac{-1}{2\hat{\sigma}^2(1 - \boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r})}\right] (\hat{\boldsymbol{y}} - \boldsymbol{x}^* \hat{\mu})^2 + \left[\frac{\boldsymbol{r}^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{x}\hat{\mu})}{\hat{\sigma}^2(1 - \boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r})}\right] (\hat{\boldsymbol{y}} - \boldsymbol{x}^* \hat{\mu}), \tag{3.20}$$

and – as a function of $\hat{\boldsymbol{y}}$ – measures the consistency of estimated values with observed data. By taking the partial derivative of (3.20) with respect to $\hat{\boldsymbol{y}}$ and equating to zero to maximize the likelihood, the standard Kriging estimator is

$$\hat{\boldsymbol{y}}(\boldsymbol{x}^*) = \boldsymbol{x}^* \hat{\mu} + \boldsymbol{r}^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{x}\hat{\mu}), \tag{3.21}$$

where the estimated $\hat{\boldsymbol{y}}(\mathbf{x}^*)$ could be considered as the most confident prediction given by the Kriging estimator, or in other words the predicted values are highly consistent with the limited observation data because such statistical behaviour of Kriging has a maximum likelihood value (i.e. unbiased). It is noted that the Kriging formulation is a kind of combination of PR and RBF, where the first term and second term in (3.21) are polynomial function and modified radial basis function (i.e. the basis function is replaced by $\boldsymbol{r}^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{x}\hat{\mu})$), respectively. Moreover, with going into details of (3.21), if the predicted point $\boldsymbol{x}^*$ is equal to the observed point $\boldsymbol{x}_i$, the correlation $\boldsymbol{r}^T \boldsymbol{R}^{-1}$ is equal to $\boldsymbol{I}$ (i.e. a unit matrix) and $\hat{\boldsymbol{y}}(\mathbf{x}^*)$ is equal to $\boldsymbol{x}^* \hat{\mu} + (\boldsymbol{y} - \boldsymbol{x}\hat{\mu})$, which is just equal to the observed $\boldsymbol{y}(\mathbf{x}_i)$. In other words, this could ensure that the Kriging model could pass through all the observed data, whereas PR or RBF may even misestimate the values at the already sampled points as shown in Fig. 3.1 and 3.2. Accordingly, Kriging is selected for this work as it could provide more accurate model than others.

One way of using Kriging or other surrogate methods for optimization is – iteratively selecting the infill sample to update the model and locate the global optimum. Note that the selection of the position of estimated minimum to be the infill sample may not be recommended, because it is likely being trapped at an unwanted local optimum when solving multimodal problems. The Matlab codes of using Kriging and estimated minimum to find the real optimal solution are given in Appendix A4. A set of continuous snapshots presented in Fig. 3.3 shows the whole optimization process (i.e. 8 iterations in this case) of Kriging using the estimated minimum as infill point. In this test case, the Schwefel function was used to provide 6 initial samples (i.e. the step size was set to 200) for building Kriging model, as the Kriging was expected to provide robust and accurate surrogate model using

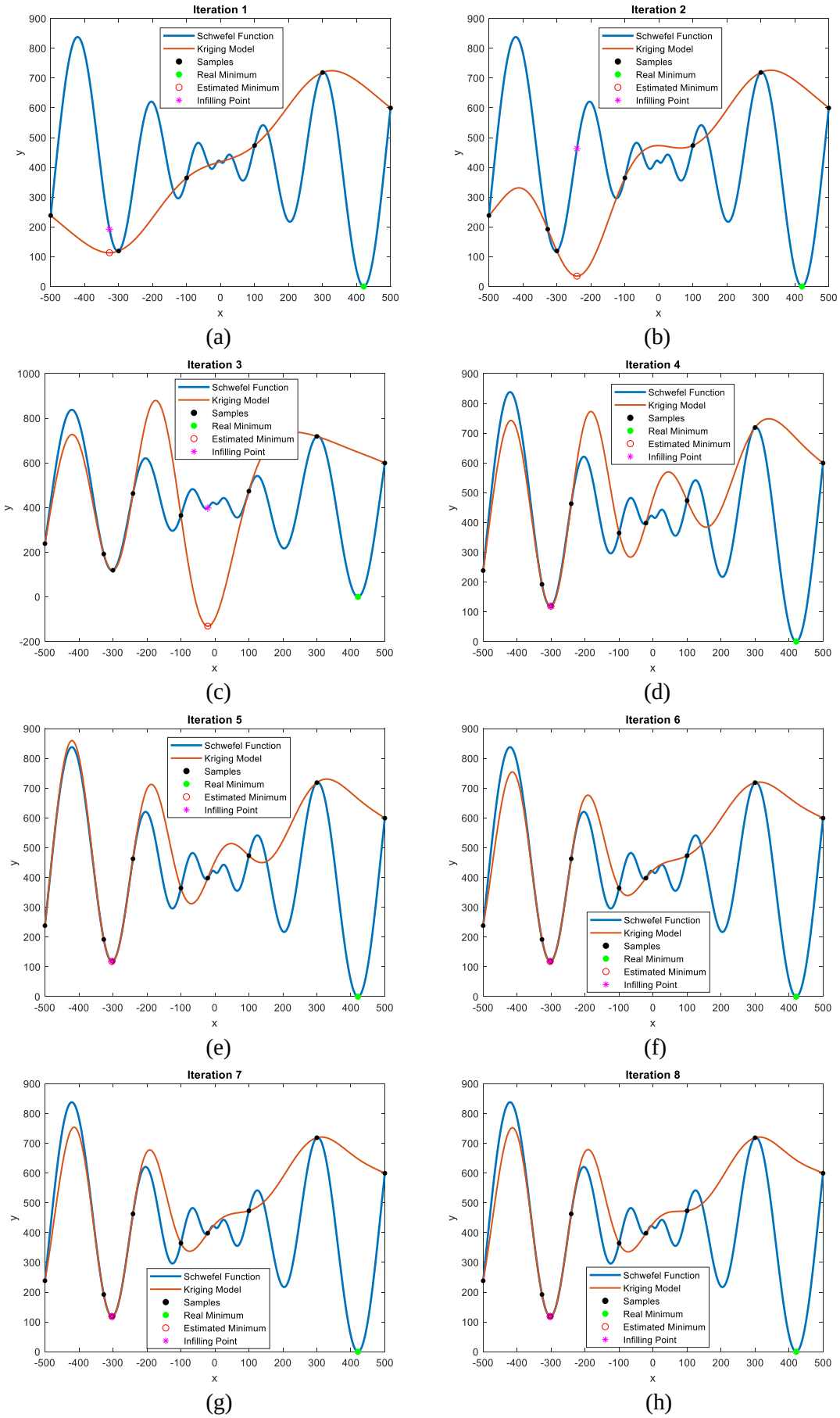


Fig. 3.3 Minimum estimation-based Kriging optimization processes (exploitative search)

small sample size. The highlighted position of estimated minimum (red circle) at each iteration is used to guide where to infill a sample (purple star) to update the model. However, it can be seen that the Kriging searches at the estimated minimum and is being trapped at the local optimum ($x = -303$, $y = 118.5$) after iteration 4. This may be caused by the aforementioned "Curse of Uncertainty" that the estimated minimum obtained from Kriging in first place is nearby a local optimum. Moreover, this is likely to happen in practical applications due to the solution region of global optimum is usually not known. On the other hand, this search strategy may be considered as fully exploitative search and it is well-suited for convex problems. In order to introduce the explorative capability to Kriging assisted optimization to jump-out the local optimum, the maximum mean square error infill criterion is presented later.

3.4.3 Mean Square Error

The motivation of using Kriging in this work is not only because it provides an accurate model to fit the observation data, more importantly, it gives the potential error to the predicted value. Note the potential error is not a true error between real value and predicted value, but an implied uncertainty of the prediction. The derived potential error named Mean Squared Error (MSE) has been given in [211] as following expression

$$\mathbf{MSE}(\boldsymbol{x}^*) = \hat{\sigma}^2 \left[1 - \boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r} + \frac{(\boldsymbol{x}^* - \boldsymbol{x}^T \boldsymbol{R}^{-1} \boldsymbol{r})^2}{\boldsymbol{x}^T \boldsymbol{R}^{-1} \boldsymbol{x}} \right], \tag{3.22}$$

The formulation (3.22) shows that the MSE indeed is defined based on the correlation matrix and variance of Kriging prediction. From statistics point view, the MSE could measure how good the predictions are and the value of MSE is inversely related to the curvature of the augmented likelihood function (3.19) as – the high curvature suggests small MSE (i.e. means high confidence of the prediction); the low curvature suggests high MSE (i.e. means low confidence of the prediction). Besides the calculated MSE is equal to zero or nearly zero if the estimated point is equal or very close to the observed data, because of the term $\boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r} = \boldsymbol{I}$ and $(\boldsymbol{x}^* - \boldsymbol{x}^T \boldsymbol{R}^{-1} \boldsymbol{r}) = 0$.

With the aid of the calculated potential error, the exploration capability of Kriging is equipped. By giving 6 initial samples obtained from Schwefel function, the whole optimization process of Kriging using maximum MSE as the infill criterion is presented in Fig. 3.4. The Matlab codes for this test are provided in Appendix A5. For better view of the plots, the obtained MSE values were normalized between 0 and 1. It is noticed that the closer the estimating point to the samples, the smaller the value of MSE (orange dash line) tend to be; and vice versa. As a result, the position of maximum MSE may exist in the middle of any two neighbouring samples. At each iteration, the point of which has maximum MSE (cyan dot) calculated using (3.22) is selected to be re-evaluated and subsequently used as an infill sample (purple star) for updating the Kriging model. It can be observed that the model is iteratively improved, which is unlike the model displayed in Fig. 3.3

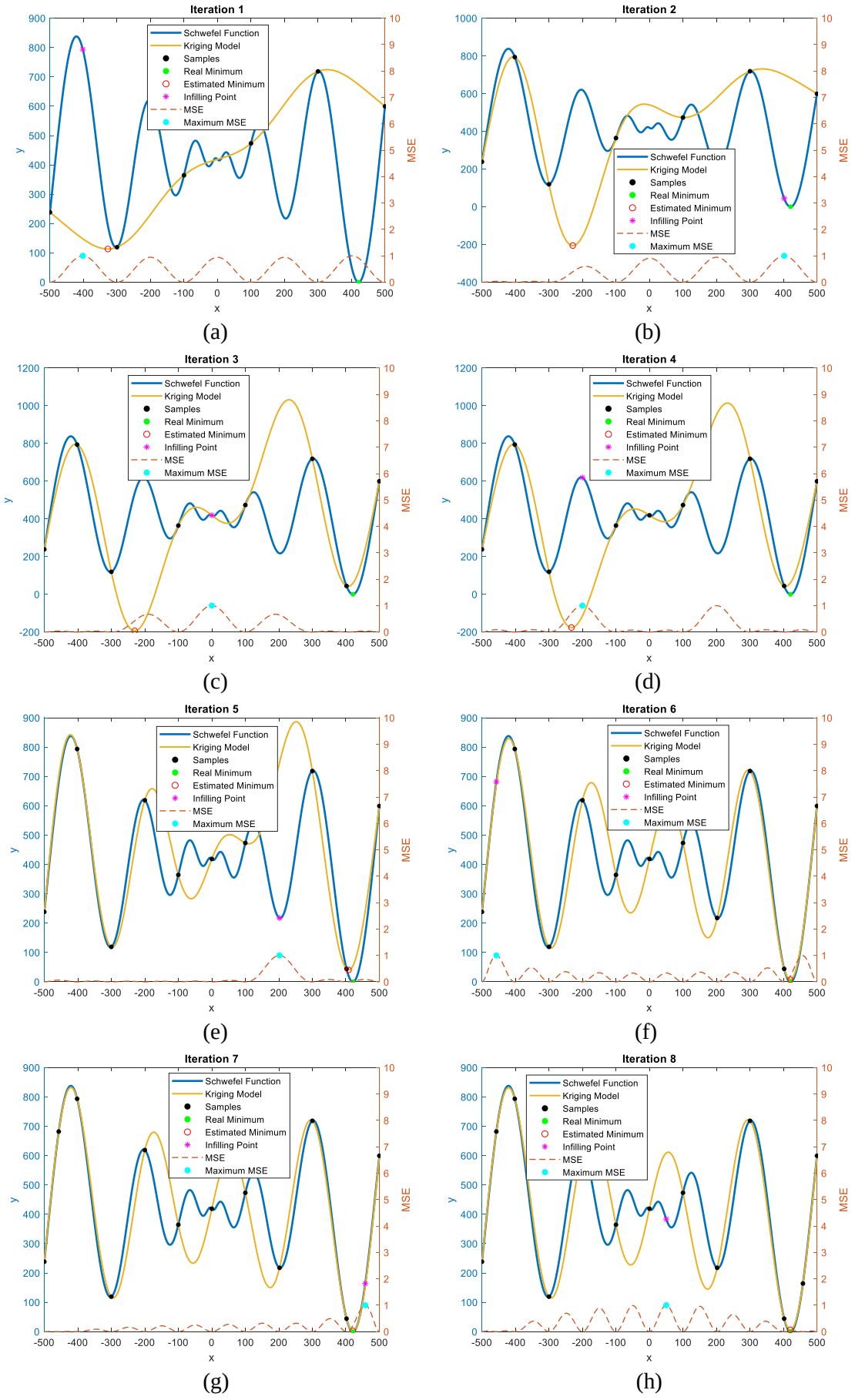


Fig. 3.4 Maximum MSE-based Kriging optimization processes (explorative search)

remaining unchanged after iteration 4. Consequently, the sub-optimal solution ($x = 402,\ y = 44.25$) nearby the global optimal ($x = 421,\ y = 0$) is found by maximum MSE at iteration 2 and the global optimum solution can be found by estimated minimum ($x = 421,\ y = 0$) at iteration 7. However, it is worthy to mention that this explorative strategy may not converge to the optimal solution unless it is associated with the previous estimated minimum strategy, because the maximum MSE point at each iteration is found to be different. To achieve this and maximize the performance of Kriging, an efficient and effective strategy considering both the estimated minimum and the maximum MSE is introduced in next chapter.

3.5 Summary

In this chapter, the fundamentals of surrogate modelling techniques including Polynomial Regression, Radial Basis Function and Kriging have been introduced. Comparing their modelling performance suggests that RBF and Kriging are capable in building accurate model using small number of samples. While these techniques associated with the infill criterions were tested on the nonlinear multimodal function to illustrate their derivative-free capability in solving nonconvex optimization problems. The test results of PR and RBF have shown that these surrogate methods using estimated minimum solution to guide the search of global optimum may be easily misled and find a local optimum, as their optimization process is a pure exploitation process without exploration. It has turned out that the PR and RBF may not be the suitable approaches for solving nonconvex optimization problems, because they are difficult to balance the exploitation and exploration. Consequently, the accurate Kriging is chosen to be the main surrogate method for this work, because the bias of local search and global search can be achieved via favouring the selection of estimated minimum and maximum MSE, respectively. In order to maximize the convergence rate while finding the global optimum when using the Kriging based optimization approach to solve the nonconvex optimization problems, the Expected Improvement infill criterion used to balance the exploitation and exploration is introduced and discussed in next chapter.

Chapter 4 Surrogate Assisted Optimization Techniques

4.1 Balance Between Exploitation and Exploration

4.1.1 Introduction

As mentioned before, the main idea of using surrogate methods to solve the optimization problems is that – using the pre-defined infill criterion to iteratively select a "most promising" point (i.e. infill point) to improve the surrogate model and approach the global optimum. Such optimization process could also be considered as a "black-box" optimization and a simple optimization flow chart may be illustrated as in Fig. 4.1. Here a question is raised – how and where to select the suitable infill point to balance the exploitation and exploration? Namely, the exploitative search (i.e. local search) refines around the current minimum to improve the solution, whereas the explorative search (i.e. global search) encourages to search away from the current minimum to locate the region of which may exist better solution. These concepts are very important for any stochastic based optimization method – the surrogate-based optimization techniques are no exception, because these could help to avoid being trapped at local optimum and fast and accurately converge to global optimum. However, the aforementioned estimated minimum and maximum MSE infill criterions are two extreme cases, which lack balance – either pure exploration or full exploitation. In other words, the local-global

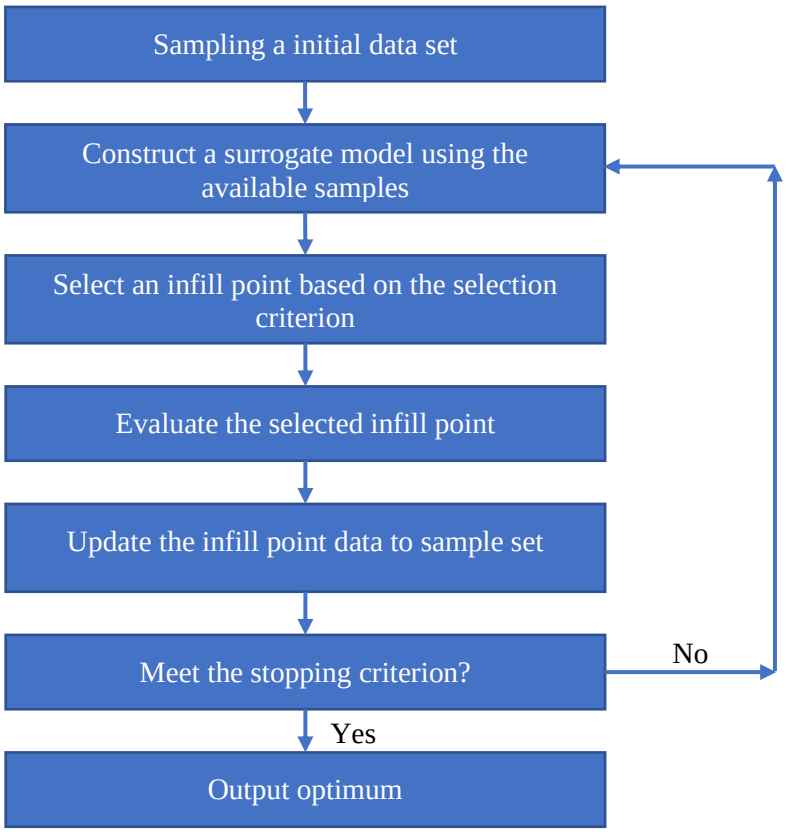


Fig. 4.1 Flow chart of the surrogate-based optimization

balance could be achieved by mixing the selection of estimated minimum and maximum MSE points. To effectively deal with this concern, [213] first proposed an infill criterion called Generalized Maximum Expected Improvement (GMEI) and introduced an tuneable integer parameter "g" with $g \geq 0$ to control the balance between two trends (i.e. low estimation value and high uncertainty of the estimation). By setting a value to g and calculating the genialized Expected Improvement (EI) value, this approach would suggest that a point, which has generalized maximum expected improvement over the current minimum objective value, should be selected as the infill point for next evaluation. Thus, it is flexible and easy to control the scope of search to some extent. Such as the larger the value of g the more emphasis on explorative search tends to be while the smaller the value of g the more emphasis on exploitative search tends to be. Nevertheless, it was argued in [214] that, in some cases the upper bound of g is very difficult to be defined to make the explorative bias of search, because the extreme exploration case is impossible to set an infinite value to g. Therefore [214] introduced a weight factor to standard EI equation, and thus developed a more robust way, so-called Weighted Maximum Expected Improvement (WMEI) criterion, to address the shortcomings of GMEI. The fundamental details of WMEI is introduced and studied below.

4.1.2 **Weighted Maximum Expected Improvement**

As introduced before, the Kriging predictor considers the prediction as being normally distributed with mean value $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ and variance MSE($\boldsymbol{x}^*$) obtained from the likelihood function of observation data. In other word, the prediction value could be considered as a random variable and the output $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ has maximum likelihood. Therefore, the prediction at $\boldsymbol{x}^*$ is likely to vary within some range (e.g. $\hat{\boldsymbol{y}}(\boldsymbol{x}^*) \pm \text{MSE}(\boldsymbol{x}^*)$), and it is possible to achieve an improvement $\boldsymbol{I}$ over currently observed minimum (i.e. y_{min}) when evaluating $\boldsymbol{x}^*$. Under these assumptions and for the sake of optimization, a statistical model could be used at any given point to measure its amount of improvement over y_{min} and its likelihood of achieving the improvement. While the amount of improvement of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ over y_{min} could be given as

$$\boldsymbol{I}(\boldsymbol{x}^*) = y_{min} - \hat{\boldsymbol{y}}(\boldsymbol{x}^*), \tag{4.1}$$

Mathematically, the smaller the prediction value of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ than y_{min} the larger the amount of improvement over y_{min} tends to be, and vice versa. Obviously, the predicted minimum point also has the maximum improvement value if directly using (4.1) to calculate the improvement value. However, this is too optimistic and the real value at $\boldsymbol{x}^*$ may be actually larger or smaller than current minimum due to the prediction error. It is no difference with the previous studied estimated minimum criterion if using this information to select the infill point. Moreover, it is impossible to obtain the true improvement value before objective function evaluation unless the surrogate model has very high-fidelity degree. Nevertheless, the amount of improvement of each predicted point could be derived statistically. From statistical perspective, the likelihood of achieving the improvement could

be expressed by a normal distribution function with mean value $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ and variance MSE($\boldsymbol{x}^*$) as follow

$$\frac{1}{\sqrt{2\pi}\hat{\boldsymbol{s}}(\boldsymbol{x}^*)} exp\left[-\frac{\left(\mathrm{y}_{min}-\left(\boldsymbol{I}(\boldsymbol{x}^*)+\hat{\boldsymbol{y}}(\boldsymbol{x}^*)\right)\right)^2}{2\hat{\boldsymbol{s}}^2(\boldsymbol{x}^*)}\right], \tag{4.2}$$

where $\hat{\boldsymbol{s}}^2(\boldsymbol{x}^*)$ is the MSE value obtained by (3.22), $\hat{\boldsymbol{s}}(\boldsymbol{x}^*)$ is the Root Mean Square Error $\sqrt{\hat{\boldsymbol{s}}^2(\boldsymbol{x}^*)}$ (which itself reflects the uncertainty of estimated value). The term $\boldsymbol{I}(\boldsymbol{x}^*) + \hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ derived from (4.1) may be interpreted as an improved prediction that could be used to quantify the likelihood of point $\boldsymbol{x}^*$ achieving the improvement over y_{min}. The value of Expected Improvement can be found by integrating the distribution function (4.2) as follow

$$E[\boldsymbol{I}(\boldsymbol{x}^*)] = \int_0^{\infty} I(\boldsymbol{x}^*)\left\{\frac{1}{\sqrt{2\pi}\hat{\boldsymbol{s}}(\boldsymbol{x}^*)} exp\left[-\frac{\left(y_{min}-\boldsymbol{I}(\boldsymbol{x}^*)-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)\right)^2}{2\hat{\boldsymbol{s}}^2(\boldsymbol{x}^*)}\right]\right\} d\boldsymbol{I}(\boldsymbol{x}^*), \tag{4.3}$$

which can be re-formulated as [74]

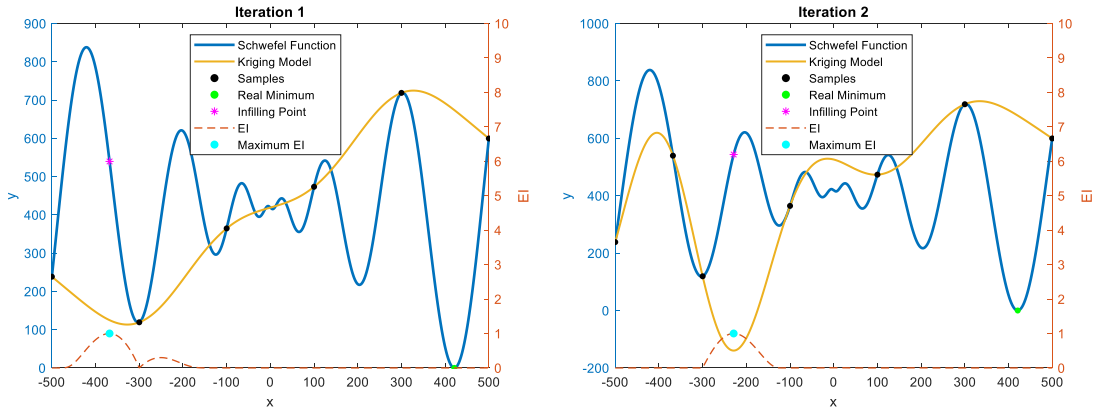
$$E[\boldsymbol{I}(\boldsymbol{x}^*)] = \begin{cases} [y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)]\Phi\left(\frac{y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right)+\hat{\boldsymbol{s}}(\boldsymbol{x}^*)\phi\left(\frac{y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right) & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*)>0 \\ 0 & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*)=0 \end{cases}, \tag{4.4}$$

where Φ is the Cumulative Distribution Function (CDF) of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ that reflects the cumulative probability of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ small than y_{min} and ϕ is the Probability Density Function (PDF) of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ that tells the probability density of $\hat{\boldsymbol{y}}(\boldsymbol{x}^*)$ equal to y_{min}. The equation can be interpreted as follows: the first term is large when Kriging prediction is likely smaller than current minimum, whilst the second term is larger when Kriging prediction is with high uncertainty (large MSE). The calculated EI could also be considered as a compromise between exploitation (first term) and exploration (second term), as both the estimation and its uncertainty are considered. The Matlab codes of WMEI are given in Appendix A6.

In order to define the preference of local search and global search when selecting the infill point, the weight factor could be introduced into (4.4) and the Weighted Maximum Expected Improvement may be expressed as follow

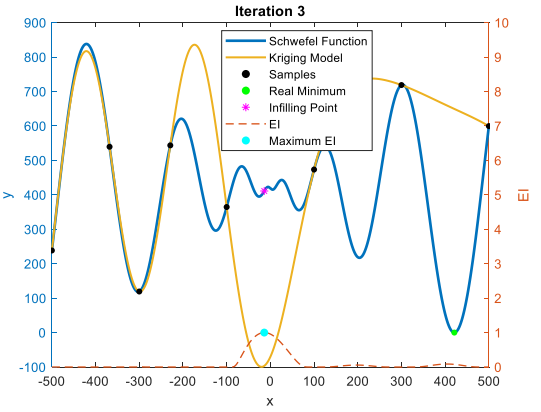
$$E[\boldsymbol{I}(\boldsymbol{x}^*)] = \begin{cases} w[y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)]\Phi\left(\frac{y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right)+(1-w)\hat{\boldsymbol{s}}(\boldsymbol{x}^*)\phi\left(\frac{y_{min}-\hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right) & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*)>0 \\ 0 & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*)=0 \end{cases}, \tag{4.5}$$

where $w \in [0,1]$, $w = 1$ and $w = 0$ encourage full exploitation (i.e. depends only on the prediction value) and pure exploration (i.e. depends only on the prediction uncertainty), respectively. Therefore, the trade-off between the predicted minimum and the maximum MSE could be achieved by properly setting the weight w to EI. As studied in section 3.4.3, the MSE value is zero if the predicted point

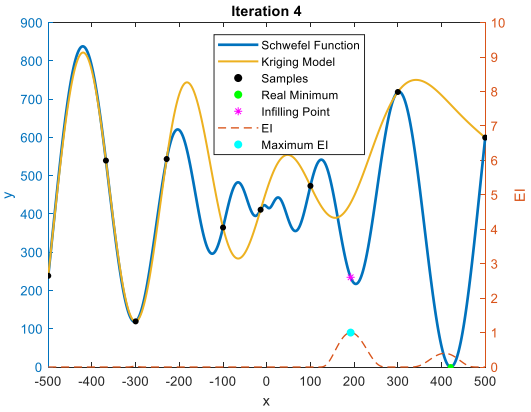


(a)

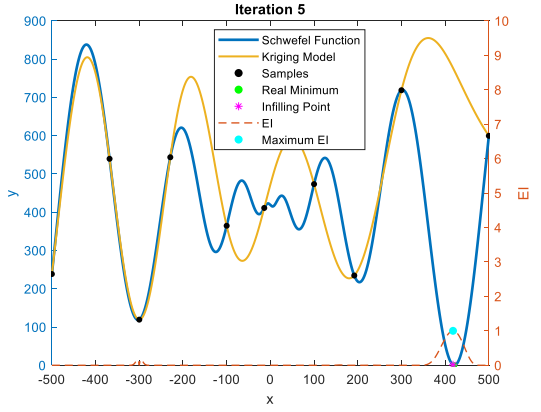
(b)



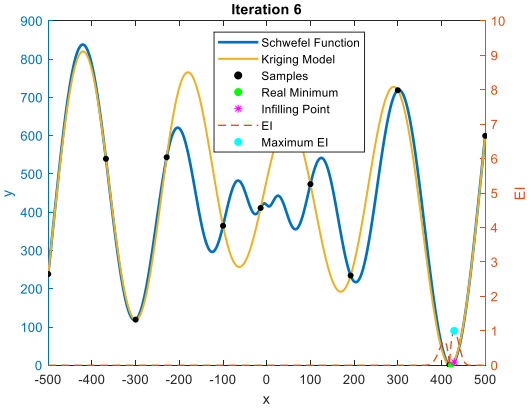
(c)



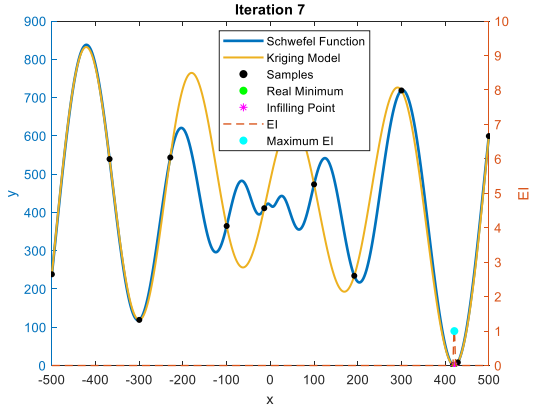
(d)



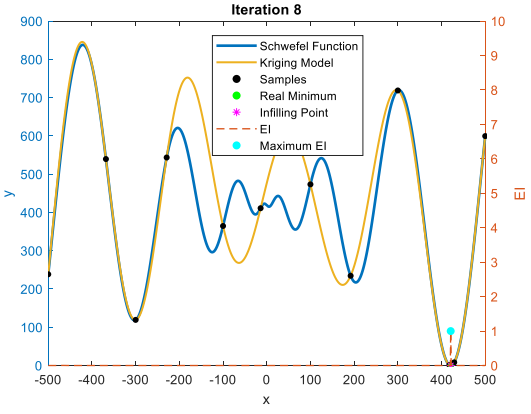
(e)



(f)



(g)



(h)

Fig. 4.2 WMEI-based Kriging optimization process (balanced search)

is same with the already sampled point. Consequently, such point cannot contribute to improve the current minimum solution, as its EI value is set to zero. Fig. 4.2 shows the iterative optimization procedures of the Kriging using WMEI infill criterion with weight setting $w = 0.5$. Note that this test is similar to the previous PR, RBF and Kriging tests, except that the infill criterion (i.e. WMEI) is different. It can be observed from Fig. 4.2 that the EI values (orange dash line) around the globally optimal solution region after iteration 4 are visibly larger than other regions and thus the infill points (purple star) at iteration 5 ($x = 418$) and iteration 6 ($x = 429$) suggested by WMEI (cyan dot) are being around the global optimum (i.e. $x = 421$). Moreover, the infill points selected after iteration 6 are exactly the same with the global optimum (i.e. converged) due to the WMEIs are found at $x = 421$. Furthermore, it was noticed that WMEI criterion with weight setting 1 and 0 is equivalent to the estimated minimum and the maximum MSE infill criterions, respectively, because they presented same Kriging curve and selected same infill point at each iteration. Note that this test used equal weight (i.e. $w = 0.5$) was only to show that the WMEI criterion is capable to jump-out the local optimum and fast converge to the global optimum. For further analysis of the impact of weight factor w on converging to the global optimum, Kriging using WMEI criterion with different values of w was tested on the Schwefel and Griewank functions. The maximum iteration of the tests was specified to 20 and the Griewank function given in [204] is expressed as

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{\boldsymbol{x}_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{\boldsymbol{x}_i}{\sqrt{i}}\right) + 1 \qquad \boldsymbol{x} \in [-10, 10], \tag{4.6}$$

where d is the number of dimensions. Besides, Kriging used 7 samples with step size 3 to model Griewank function. The Matlab codes of this test can be found in Appendix A6. Table 4.1 presents the number of iterations required to find the global optimum when the value of weight factor w

Table 4.1 The performance of finding global optimum with different weight factor values

	Griewank Function	Schwefel Function
Value of weight	The number of iterations required to	The number of iterations required to
1	failed	failed
0.9	failed	12
0.8	failed	12
0.7	failed	12
0.6	4	8
0.5	5	7
0.4	5	8
0.3	6	7
0.2	4	7
0.1	6	7
0	sub-optimum	failed

* Note: The "failed" means that the Kriging with specified value of weight fails to converge to the global optimum. The "sub-optimum" means the obtained optimal solution is very close to the exact global optimum.

decreasing from 1 to 0 (with step size 0.1). For both test functions, the results indicate that the extreme cases (i.e. $w = 0$ and $w = 1$) are not the good settings for solving nonlinear optimization problems and it is recommended to set equal weight or put more emphasis on the explorative search (i.e. smaller value of w) to EI. Therefore, the value of w has to be carefully defined for different optimization problem, while the constant weigh factor may not be favoured in practical optimization problems. To achieve best convergence performance, the weight may need to be adaptively adjusted along with the optimization process. Under this concern, the adaptive weight strategies, which could automatically adjust the weight value used in WMEI during the optimization processes, are proposed and introduced in later chapter.

4.2 Efficient Global Optimization

4.2.1 Introduction

The use of Kriging associated with maximum expected improvement to solve optimization problems is named Efficient Global Optimization (EGO) as proposed in [74]. This technique in fact is a Bayesian-based optimization, which uses the statistical sense and all available samples to predict which point is worth to be evaluated in the next run. For low-dimensional optimization problems, it is straightforward to perform exhaustive search that predicts all design points within the search space to find the maximum EI point (e.g. 1001 points with step size 1 were taken in previous tests). However, the computational explosion will occur if the problem becomes high-dimensional. For example, if a 10-D problem considers 100 samples in each dimension, 10^{20} possible combination points need to be calculated to determinate the maximum EI among them, requiring unaffordable computational cost. A large step size or small number of samples may be easier to handle, but the maximum EI point may be overlooked as well as the global optimum. Therefore, the EGO treats the task of finding maximum EI point as an optimization problem, which could be solved by many optimization solvers. In this way, the EGO can be considered as a Kriging based optimization method assisted by another optimization solver. When Kriging model is combined with the evolutionary computation, technically, the optimization of WMEI may require less computational efforts than randomly searching the optimum within the solution space. It could potentially prevent too much function evaluations and improve the computational efficiency as only the suggested “promise” point is evaluated using real objective function. Nevertheless, the weighted EI usually presents nonlinear and nonconvex behaviour and it iteratively changes along with the optimization process and the updated surrogate model (e.g. the EI curve shown in Fig. 4.2). As a result, the global WMEI may not easy to be found by the adopted optimization solver and the EGO may not find the desirably global optimum. Moreover, the application of EGO so far has been limited to relatively small-scale

optimization tasks because of the computational cost issue caused by correlation matrix inversion [215], whereas OPF problems tend to have a large number of variables.

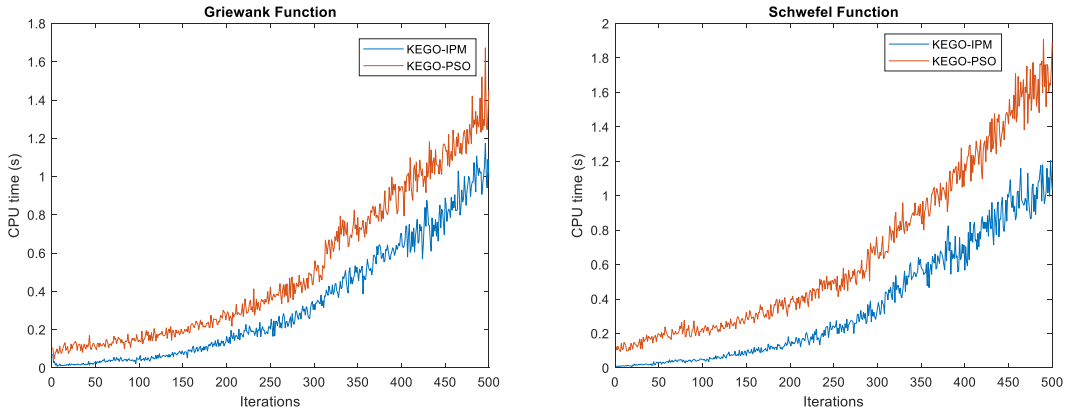
4.2.2 **Numerical Test of EGOs**

To verify the performance of EGO, two Kriging Efficient Global Optimization (KEGO) algorithms assisted by IPM and PSO (i.e. KEGO-IPM and KEGO-PSO), respectively, were implemented (by integrating the Kriging with Matlab's inner IPM and PSO functions) and tested. The related Matlab codes can be found in appendix A7. For this test, the Schwefel function and the Griewank function with 5 variables were considered as the optimization problems, 50 random initial samples uniformly distributed within specified range were generated to build the initial Kriging model, the start point of IPM was also uniformly random generated, the population size and maximum generation of PSO were set to 10 and 100, respectively, the equal weight of EI was used (i.e. w used in WMEI was set to 0.5), and the maximum iterations of EGO was set to 500. Note these parameter settings are not the fine-tuned values, the accurate optimal solutions for the test functions are all equal to zero, and the variables of Griewank function is limited between -600 and 600 in this case.

The optimal solutions in terms of worst, best, mean and Standard Deviation (std) values of the 20 trials obtained from KEGO-IPM and KEGO-PSO are presented in Table 4.2. These obtained results indicate that the local optima of Schwefel and Griewank functions are found by both algorithms, although the solutions provided by the KEGO-PSO are smaller. However, the KEGO assisted by PSO spent much more computation time than KEGO-IPM. These are the not uncommon features when using local optimizer and global optimizer to deal with the nonlinear optimization problems – in this test the IPM guarantees a local WMEI when selecting infill point and the PSO requires more computational efforts to find the global or local WMEI. Furthermore, the average CPU times of both algorithms dramatically increased at each iteration, as plotted in Fig. 4.3. It is mainly caused by the iteratively increased sample and hence needed to invert a large correlation matrix associated with $O(n^3 d)$ computational cost [76], where n is the number of samples and d is the number of variables. This issue has limited the application of EGO to small-scale optimization problems. The EGO may be not a suitable technique for solving OPF problems (or improving the performance of evolutionary

Table 4.2 Worst, best, mean and standard deviation solutions of the 20 trials

Test Functions	Algorithms	Worst	Best	Mean	std	Average CPU time (s)
Griewank Function	KEGO-IPM	17.082	1.2715	8.8533	4.8407	165.68
	KEGO-PSO	3.7014	0.7734	1.7120	0.6717	255.29
Schwefel Function	KEGO-IPM	1048.2	441.61	880.40	159.12	181.35
	KEGO-PSO	1016.3	399.29	777.96	152.55	334.10



(a) (b)

Fig. 4.3 Average computation times of KEGO-IPM and KEGO-PSO at each iteration

computation) that contain dozens or hundreds of variables or more, due to the process of finding WMEI at each iteration may require more computational efforts than power flow calculation. Consequently, the EGO technique is not studied and discussed further in this work, although it is somewhat new to the OPF problems. Nevertheless, it is worthy to mention that the EGO is still capable of improving the computational efficiency of those very "expensive" optimization problems requiring hours or more to evaluate the objective function, because the iteratively increased CPU time at each iteration is comparably negligible. In next section, a more promising and efficient approach, which uses surrogate modelling method to assist and improve evolutionary computation, is introduced and reviewed.

4.3 Surrogate Assisted Evolutionary Computation

4.3.1 Introduction

Evolutionary computation is very robust and flexible to solve those nonlinear and nonconvex OPF problems of which deterministic methods experiencing difficulties in finding global optimum. This benefits from their derivative-free and global search features. Nevertheless, the evolutionary based algorithms usually require enormous function evaluations that have limited their application in practical OPF problems, especially, when the computational budget is restricted (e.g. real-time OPF). To deal with this concern, the surrogate modelling techniques could be used in this work. However, the way of using evolutionary computation to assist surrogate modelling method (i.e. EGO) may not be computationally efficient when solving the "cheap" OPF problems, due to the task of optimizing WMEI is nontrivial, as mentioned in previous section. By contrast, it is found that an approach called Surrogate Assisted Evolution Computation (SAEC) is more promising and reliable. Namely, this technique uses surrogate method to assist evolutionary computation and it was originally developed

for improving the computational efficiency of evolutionary computation [73]. With the aid of surrogate model, the fitness can be estimated rather than evaluated using real objective function, consequently, the total number of function evaluations is reduced as well as the computation time. This SAEC technique has been widely applied to solve those "expensive" design optimization problems, such as electromagnetic design [215], [216] and aerodynamic design [217], [218]. However, it has not been studied in the field of power system optimization and it is worthy to be investigated. Note that the design optimization problems usually consider only few design parameters (i.e. control variables) while their objective function evaluation is highly time-consuming (e.g. hours), by contrast, the OPF problems contains dozens or hundreds or even more variables and the calculation of OPF objective function may only take seconds or minutes. As a result of directly applying SAEC technique to OPF problems, the "Curse of dimensionality" [75] arises and causes significant difficult in building an accurate surrogate model for large-scale OPF problems; besides, the estimation of fitness may spend more computation time than the evaluation of OPF objective function.

Therefore these two issues (i.e. accuracy and computational efficiency of surrogate model) must be properly addressed when using surrogate method to assist evolutionary computation to solve OPF problems, otherwise the computational efficiency of evolutionary computation may be degenerated. Moreover, in order to effectively use the Surrogate assisted Evolutionary Computation to solve OPF optimization problems, it is important to answer which surrogate method should be selected and how to combine surrogate method with evolutionary based algorithm. In previous chapter, three well-known surrogate modelling methods have been introduced and studied. The Kriging associated with Maximum Expected Improvement criterion has been selected as the main surrogate method in this work, due to the distinctive property in which the local search and global search can be easily tuned. In this section, the strategies of managing surrogate method in evolutionary computation are briefly reviewed and discussed to identify and develop the suitable one for efficiently solving OPF problems.

4.3.2 An Overview of Managing Surrogate Method in Evolutionary Computation

Due to the technical limitations of EGO, this work exploits the Kriging to assist evolutionary computation to improve its computation efficiency, rather than using evolutionary computation to assist Kriging as in EGO. Nonetheless, the WMEI criterion used in EGO could be adopted in this work to balance the exploitation and exploration, and it could effectively mitigate the negative impact of prediction errors introduced by low-fidelity surrogate model on the computational performance. There are several different strategies have been reported in the literature for managing surrogate model in evolutionary based algorithm. According to the review works [73], [219]–[221], the mechanisms could be generally classified into two categories: 1). Evolution Control and 2). Pre-Selection, which are simply illustrated in Fig. 4.4 and 4.5, respectively. Moreover, the advantages and disadvantages of these two main strategies may be summarized as given in Table 4.3. By means

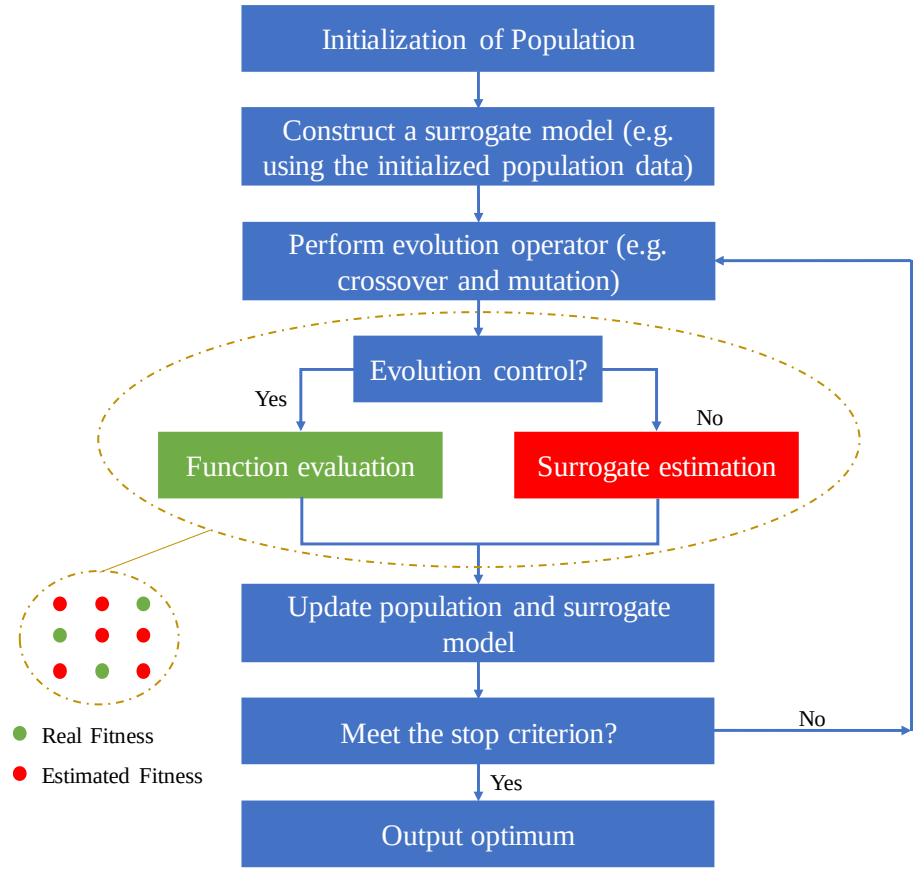


Fig. 4.4 Example of evolution control strategy

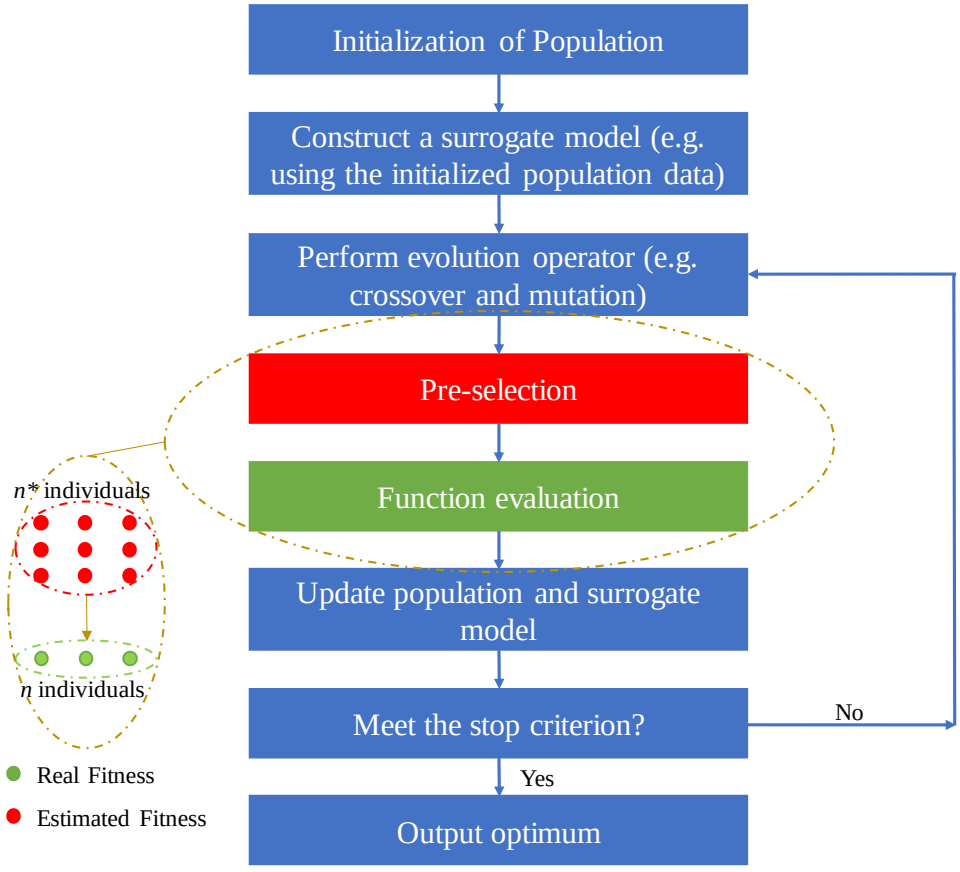


Fig. 4.5 Example of pre-selection strategy

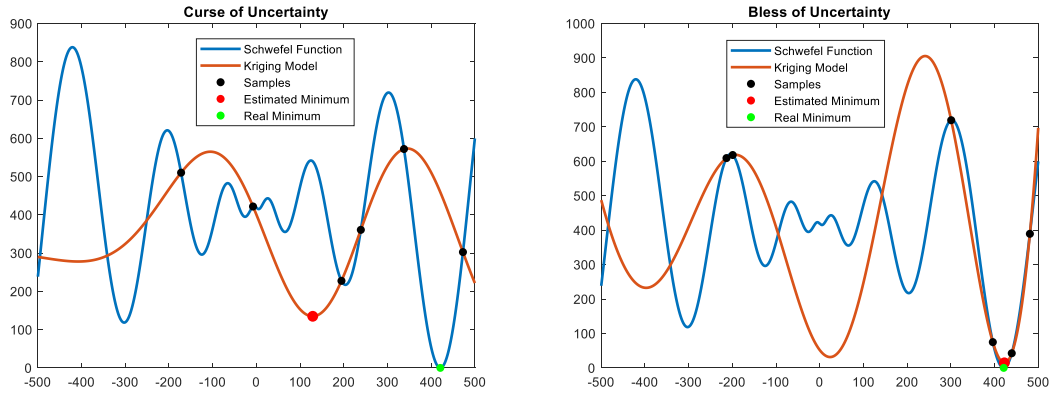
Table 4.3 Pros and Cons of different categories of managing surrogate model in the evolutionary algorithm

Category	Pros	Cons
Evolution Control	*NEC*: Most computationally efficient; particularly good for low-dimension problems.	*NEC*: Requirement on the fidelity of surrogate model is most restricted; it most likely converge to a false optimum.
	FEC & *AEC*: Hybrid use of surrogate model and original function makes it capable for solving high-dimensional problems.	*FEC* & *AEC*: It is not easy to define the suitable parameter to manage the use of surrogate model and original function.
Pre-selection	The requirement of modelling quality is minimum; the surrogate can be used to refine the local optimum; it avoids the convergence to the false optimum.	Comparing with evolution control, it is most computationally expensive.

of evolution control strategy, in the evolutionary optimization, the original function is used to evaluate the fitness of some selected individuals or all individuals at some specified generations (i.e. iterations). As a result, the fitness values of individuals in the updated population consist of both estimated fitness and real fitness. This kind of strategy has been argued that the computational cost could be reduced as much as possible [219], however, it is very likely to find a false optimum because of the difficulties in constructing high-fidelity surrogate model. In the pre-selection strategy, the surrogate method is used to screen-out the most promising individuals to be evaluated using real objective function. Although the efficiency improvement of using pre-selection strategy is deemed much lower than using evolution control [220], [221], it is much less sensitive to the degree of fidelity of surrogate model and thus reduces the possibility of misleading by surrogate modelling errors.

4.3.2.1 **Evolution Control Strategy**

No Evolution Control (NEC): The first research work on Surrogate Assisted Evolutionary Computation may be traced back to 1980s [222], which analysed and demonstrated the effectiveness of using linear regression estimation on improving the efficiency of Genetic Algorithm by the experiment on Image Registration. This perhaps the most straightforward approach of using surrogate method in evolutionary optimization. It assumes that the approximation model is globally correct at which the position of estimated optimum is the real optimum or at least a near-optimum, and it suggests to use the estimated fitness instead of real fitness for individual evolutions during the whole optimization process (i.e. no evolution control). However, this approach, similar to the surrogate optimization using estimated optimum to find the global optimum, may provide a false optimum and it is unreliable when dealing with complex optimization problems. Because it is very difficult to build a globally correct surrogate model, especially when the samples are limited and the problems have many variables. As shown in Fig. 4.6 (a), the evolutionary algorithms will converge



(a) (b)

Fig. 4.6 Curse and blessing of uncertainty when using surrogates in evolutionary optimization

to the false optimum (i.e. the estimated minimum represented by the red dot) if the individual evolution fully rely on the estimated fitness; whereas the evolutionary algorithm will at least find a near-optimum solution because the estimated minimum given by the Kriging model is very close to the real optimum (green dot), as plotted in Fig. 4.6 (b). Moreover, these concerns have been termed as – "Curse of Uncertainty" and "Blessing of Uncertainty", respectively, in [75].

Fixed Evolution Control (FEC): In order to avoid the evolutionary algorithm from being misled by the false optimum introduced by surrogate model, many works have suggested that the surrogate model should be used together with the real objective function. For example, [223] investigated the method of which the best individual (i.e. better estimated fitness than others) in every generation is evaluated using real objective function, [224] used a convergence criterion to determinate when the original function must be used to evaluate the fitness and update the surrogate model; [225] proposed that the surrogate model should be employed for k generations, and then in the $(k+1)^{\text{th}}$ generation the surrogate function is replaced by the original function for function evaluations and updating the approximate model; [226] introduced that the estimated fittest individual in the population should be evaluated using explicit fitness function every R generations for retraining the surrogate model. Their simulation results have shown great improvement on the solution quality when the surrogate method used in evolutionary computation is properly managed. The strategy of surrogate model management in evolutionary computation has been conceptualized as "Evolution Control" in [227], where two generalized methods of combining the estimated fitness with the real fitness have been identified and summarized. The first one is named controlled individuals, which means some selected individuals are controlled to be evaluated using real function at each generation; the second one is called controlled generation, which the real function is used to evaluate all individuals every M generation.

Adaptive Evolution Control (AEC): In order to make the evolution control more adaptive, [228] developed a fuzzy system based method to define which generation should use the real function to evaluated the fitness and update the surrogate model. The test results obtained from 12-D and 20-D

Ackley and Rosenbrock functions have shown that the solution quality and the computation time of evolution strategy – Covariance Matrix Adaptation algorithm – are improved by using Neural Network model with proposed adaptive generation-based evolution control strategy, but the obtained solutions are in fact neither the global optimum nor the near-optimum.

Although the Fixed Evolution Control and the Adaptive Evolution Control somehow could mitigate the "Curse of Uncertainty" and benefit from the "Blessing of Uncertainty", the required high-fidelity surrogate model remains a very challenging research topic, especially for large-scale optimization problems. Moreover, the parameters used to alternate between surrogate model and original objective function is not easy to be defined, this is particularly true when the computational budget is limited and a small number of samples are used to build the surrogate model. In later chapter, a relatively conservative individual-based strategy without considering approximated fitness in individual evolution process is proposed to prevent the algorithm from being misled by false optimum introduced by surrogate model and mitigate the requirement of high-fidelity surrogate model when solving OPF problems.

4.3.2.2 **Pre-selection Strategy**

In addition to use the Evolution Control strategy to manage the use of surrogate model and real objective function during evolutionary optimization process, the surrogate model can also be applied to evolution operator (e.g. crossover and mutation) to guide or inform the initialization and update of population. This strategy was first introduced in [229] and termed as informed operator (also known as pre-selection). Such as in [230], a 2nd order Polynomial Regression model has been proposed to guide the generation of search point of offspring, rather than randomly generating offspring through crossover and mutation; [231] proposed to use a local optimizer to minimize the constructed Support Vector Machine model to find the potential best mutation. The general idea behind this pre-selection strategy is to generate a set of candidate solutions using crossover or mutation or other evolution operators, and then estimating them via surrogate model in order to select the most promising solution (e.g. the solution with best approximate value) for real objective function evaluation. Because the solutions of evolutionary based algorithm are usually randomly generated by the operators, the use of low-fidelity surrogate model to screen-out the solutions (i.e. select the good solutions and filter the poor solutions) is intuitively better and may not be worse than carrying them out randomly (i.e. reduce the randomness of operators). On the other hand, due to the estimated fitness is not directly used to evolutionary optimization process, the pre-selection strategy is regarded as less sensitive to model fidelity and is capable to prevent the risk of converging to false optimum offered by surrogate model. In addition to the pre-selection strategy, recent works in [232], [233] proposed that the fitness of a particle will be evaluated using real objective function only if the estimated fitness of this particle is less than its real fitness of personal best, otherwise the process of particle evaluation is skipped. The effectiveness of this kind of pre-selection strategy, achieving reduction on the number of function evaluations at each generation, has been demonstrated by the

benchmark optimization function tests, however, there are two potential drawbacks should be concerned. First the optimum solution may be overlooked, because the comparison between estimated fitness and personal best is used to decide whether the particle is evaluated using real objective function. Second there is no difference between original PSO and this proposed surrogate assisted PSO if large estimation errors exist that the estimation of all particles are all smaller than their personal best, which means the pre-selection strategy is failed. In other word, this strategy may still require a high-fidelity surrogate model in evolutionary optimization.

Overall, the pre-selection may be intuitively suitable for solving OPF problems, because it could potentially address the drawbacks presented in evolution control strategy. Nevertheless, this technique may increase the computational cost of evolutionary based algorithm, because the additional efforts are required before each objective function evaluation to filter out the unwanted solutions. To minimize the computational cost accompanied by surrogate model, the local Kriging model built by fixed number of samples is proposed to refine and improve the solution generated by evolutionary operator in later chapter.

4.4 Summary

In first two sections of this chapter, the Weight Maximum Expected Improvement criterion and the EGO have been introduced and tested on the numerical optimization functions. It has been found that the trade-off between local search and global search could be easily adjusted via tuning the weight factor w. However, the optimal value of weight factor is problem dependency and the weight value has significant impact on the convergence performance of Kriging optimizer. A large value of w puts excessive emphasis on exploitation and has excellent convergence speed, but it is too risky and the final solution may be trapped at local optimum; whereas a small value of w encouraging exploration is helpful to avoid local optimum, but it requires more function evaluations and thus may be degenerating the computational efficiency. For this reason, an adaptive strategy has to be developed in following chapters to tune the weight w automatically along with the optimization process. Moreover, it has been demonstrated that the use of Kriging as EGO undergoes $O(n^3 d)$ computational cost when dealing with optimization problems. Furthermore, the EGO using an additional optimization solver to find the WMEI may not be efficient than directly optimizing the "cheap" optimization problem. Therefore, the EGO is not considered in this work to solve the relatively "cheap" OPF problems containing many control variables, but it does drive this work to develop an effective strategy to alleviate the computational burden of building Kriging model associating with sample size.

In the last section, it has been discussed that the SAEC technique is a more robust and reliable in solving OPF problems than the EGO, because the SAEC could reduce the number of necessary

function evaluations of evolutionary computation as well as the computational cost. Nevertheless, the use of surrogate model and real objective function should be properly managed when using the SAEC technique. The pros and cons of two main categories of managing surrogate model in evolutionary optimization have been briefly reviewed and discussed. It has been found that those strategies are developed for solving the small-scale and computationally expensive optimization problems, thus the SAEC cannot be applied straightforward to solve OPF problems. Moreover, it is difficult to make the decision on which strategy is the best for solving OPF problems, since they have not been applied to the field of power system optimization before. Nonetheless, it could be concluded that the issue of converging to false optimum and requirement on accurate surrogate model has to be properly addressed or alleviated in order to use the evolution control based strategy; whereas to use the pre-selection based strategy, the computational cost concern has to be properly dealt with. In later chapters, two improved strategies inspired from control and pre-selection, respectively, while addressing these concerns are proposed to improve the computational efficiency of evolutionary computation when solving the concerned OPF problems.

Chapter 5 Development and Implementation of Individual-based Kriging Assisted Evolutionary Algorithms

5.1 Introduction

Amongst various evolutionary based algorithms, GA and PSO are best known with proven computational performance in wide range of applications. However, there are many variants of GA and PSO available in the literature, and it is difficult to make the decision of which variant is the best one, as they may present different performances in solving different OPF problems (i.e. "No Free Lunch Theorem" [234]). Nevertheless, it could be deemed that those variants have developed for the sake of advancing the canonical GA and PSO or addressing their drawbacks. Therefore, without loss of generality and increasing the complexity of algorithm, the proposed strategy in this work is to combine the Kriging with the canonical version of GA and PSO. On the other hand, in order to deal with the continuous variable, Real-Coded GA (i.e. using floating number or real-valued to represent the variables) is employed rather than Binary-Coded GA. Using binary with fixed string length to represent the continuous variable usually results in sacrifice of precision. Although the precision could be improved by extending more bit strings, the length of binary individual would become longer (especially large dimension is considered), while the nontrivial encoding-decoding process requires more computational efforts, considerably slowing down the algorithm, as argued and proven in [235], [236]. Moreover, the Real-Coded and Binary-Coded GA were tested on several multimodal benchmark optimization functions to illustrate their computational efficiency, as presented in the appendix B1 (e.g. using binary code requires 20 times more computation time than using real code in the worst case). Besides, the Real-Code algorithm can easily handle the constraint of control variables by directly enforcing them to their specified range. Under these concerns, therefore the real code is decided to be applied to this work. However, it seems there is no clear definition on the canonical Real-Coded GA in the literature, and only the canonical Binary-Coded GA has been defined in [237], where uses roulette wheel selection, single point crossover and bit-flip mutation. Due to the bit-flip mutation is limited to the Binary-Coded GA, a well-performed non-uniform mutation [238] that could be able to maintain the convergent performance during the optimization process is used in this work. These GA operators are introduced and discussed in later section. Finally, the canonical PSO defined in [180] where uses constriction coefficients and "gbest" topology is employed in this work. As introduced and discussed in chapter 2, the use of constriction coefficients and "gbest" topology in PSO is able contribute to excellent convergence performance as well as the computational efficiency. For the convenience, GA and PSO in later means the canonical version if no other statements are given.

At the very beginning of this project, it was decided to use EGO to solve the OPF problems, due to the previous successful Southampton Group works [79], [80]. However, the high computation complexity of EGO caused by the iteratively increased sample size, the correlation matrix inversion and the way of finding infill point, makes it inefficient for solving OPF problems, as discussed and studied in last chapter. Although [215], [216] proposed two approaches to alleviate the computational burden from large data set, the EGO yet cannot be directly applied to OPF applications. In addition to EGO technique, the evolution control strategy of SAEC also cannot be adopted freely, due to the difficulty in building high-fidelity surrogate model for large-scale OPF problems. Nevertheless, it has to emphasize that the well-performed EGOs and evolution control strategies throughout the literature are developed to solve those small-scale expensive optimization problems more efficiently, whereas OPF comparing with them is very "cheap" while considering many more variables. These concerns have motivated this work to develop an improved approach that inherits the advantages of using SAEC technique to solve expensive optimization problems, while efficiently solving OPF problems.

In order to effectively exploit the Kriging to reduce the computational burden associated with applying evolutionary optimization to OPF problems, an improved individual-based strategy – which efficiently mitigates the fidelity concern and the $O(n^3 d)$ computational cost – is proposed in this chapter. It is worthy to mention that the proposed strategy is inspired from both EGO and evolution control strategy that could be regarded as a trade-off between them. However, unlike the EGO using an optimizer to find the WMEI infill point, the proposed individual-based strategy only selects one individual/particle that has WMEI from the population at each iteration to evaluate its fitness using real objective function. To prevent the risk of individuals/particles being misled by the estimated false optimum as in the evolution control strategy, the proposed individual-based strategy suggests that the estimated fitness should not be involved in the evolutionary optimization process. Moreover, a Euclidean distance-based method is proposed to adaptively balance the exploitation and exploration when selecting an individual (or particle) for objective function evaluation. In addition, the effective approach of updating population (or swarm) and sample set (used to build local Kriging model and mitigate the computational cost of Kriging estimation) is proposed. Furthermore, a Gaussian based mutation operator is incorporated to avoid local optimum. Finally, to verify the proposed strategy, the Kriging Assisted Genetic Algorithm (KAGA) and the Kriging Assisted Particle Swarm Optimization (KAPSO) are developed and tested on five multimodal optimization functions with results compared with the original GA and PSO. After verifying their reliability and robustness of solving nonlinear and nonconvex optimization problems, KAGA and KAPSO are then chosen and applied to the benchmark IEEE 30 and 118 bus systems for minimizing the generation cost and active power losses. Their ensuing OPF results are compared with reported results from literature to prove the effectiveness of proposed Kriging individual-based strategy as well as the proposed KAGA and KAPSO.

5.2 Proposed individual-based Algorithms

For better view of the proposed algorithms, the pseudo-code and the flow chart of KAGA and KAPSO are presented in Fig. 5.2 and 5.3, respectively, while the related Matlab codes, used to solve the numerical optimization and OPF problems considered in this thesis, can be found in the appendix A8 and A9. The detail of developed KAGA and KAPSO associating with the proposed adaptive Euclidean distance based WMEI, improved individual-based Kriging, population/swarm updating and "better-adoption, worse-abandonment" sample updating, and elitist learning strategies are described and discussed in the follows

1) Initialization

This step defines the population/swarm size, the maximum number of iterations and the essential parameters for GA (e.g. crossover rate and mutation rate) or PSO (e.g., inertia weight, and cognitive and social learning coefficients). Then a uniformly distributed population/swarm is randomly generated consisting of a specified number of individuals/particles whose fitness is evaluated by objective function. The population/swarm later is used as the sample to build an initial Kriging model.

2) Generating offspring or updating particles

In order to save the computational budget, only one statistically good individual/particle (i.e. which has WMEI value) from the updated population/swarm will be selected in next step to retrain the Kriging model. This is claimed to be a different mechanism from the EGO using an optimizer. For GA, the roulette wheel selection, the single point crossover and the non-uniform mutation operators, are employed to generate the offspring. The probability of choosing individuals as parents for generating offspring is given as

$$\boldsymbol{p}_i = \frac{1/\boldsymbol{f}_i}{\sum_i^n (1/\boldsymbol{f}_i)}, \tag{5.1}$$

where $\boldsymbol{p}_i$ is the probability of i^{th} individual, f_i is the fitness of i^{th} individual, and n is the total number of individuals in the population. By using the roulette wheel selection, the fitted individual is more likely to be chosen for crossover, due to the population being sorted in ascending order according to their fitness value. In other words, it is more likely to keep the information of good solution and pass it to the offspring. Moreover, it is more focused on the exploitation to improve the convergence. If the parents are selected from the same individua, a different individual will be randomly re-selected from the population as a parent. Besides, the single point crossover is selected not only because it is recommended in the canonical version of GA, but also because it could retain some variables that may belong to the global optimum when reproducing offspring, finding better optimal solutions. The single point crossover is illustrated in Fig. 5.1. In addition, the non-uniform mutation is given as

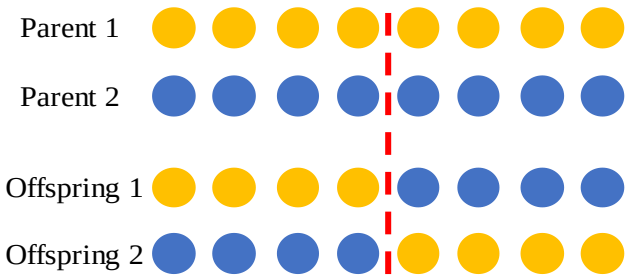


Fig. 5.1 Illustration of single point crossover (where the crossover point is randomly selected)

$$\boldsymbol{x}_{new}^{d} = \boldsymbol{x}^{d} \pm \left(\overline{\boldsymbol{x}}^{d} - \underline{\boldsymbol{x}}^{d}\right)\left(1 - r^{\left(1-\frac{FEs}{MaxFEs}\right)}\right), \tag{5.2}$$

where $\boldsymbol{x}$ is a d-dimensional vector referring to the selected offspring to be mutated, d is uniformly random selected, $\boldsymbol{x}^{d}$ and $\boldsymbol{x}_{new}^{d}$ is respectively the original and mutated values of d^{th} variable of $\boldsymbol{x}$, $\pm$ means that the positive or negative sign is selected with probability 0.5, $\overline{\boldsymbol{x}}^{d}$ and $\underline{\boldsymbol{x}}^{d}$ denote the upper and lower bounds of d^{th} variable, respectively, r is a uniform random number between 1 and 0, FEs is the current number of function evaluations, and $MaxFEs$ is the specified maximum number of function evaluations. In order to achieve a good convergence performance when using mutation operator in the GA, the non-uniform mutation is selected because it uses the term $FEs/MaxFEs$ to control the mutation step length as well as the convergence, which nonlinear decreases along with the number of function evaluations.

For PSO, the constriction coefficients are used to calculate the velocity of each particle for updating the swarm, which could be expressed as follows:

$$\boldsymbol{v}_i^{k+1} = \omega \boldsymbol{v}_i^{k} + c_1 \boldsymbol{r}_1 (\boldsymbol{P}_{best_i} - \boldsymbol{x}_i^{k}) + c_2 \boldsymbol{r}_2 (\boldsymbol{G}_{best} - \boldsymbol{x}_i^{k}) \tag{5.3}$$

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^{k} + \boldsymbol{v}_i^{k+1}, \tag{5.4}$$

where i denote the i^{th} particle, k denote the k^{th} iteration, $\omega = 0.7298$ is the inertia weight factor, $c_1 = 1.4962$ and $c_2 = 1.4962$ are cognitive and social learning coefficients, respectively, $\boldsymbol{P}_{best_i}$ and $\boldsymbol{G}_{best}$ are vectors of personal and global best position, respectively, $\boldsymbol{v}_i$ is the velocity vector of i^{th} particle, $\boldsymbol{x}_i$ is the position vector of i^{th} particle, and $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are the vectors consisting of random numbers between 0 and 1. As discussed in chapter 2, the exploitative and explorative bias of searches can be adjusted via tuning the parameters ω, c_1 and c_2 and the swarm topology. The constriction coefficients and the "gbest" topology are used in this work, because they are helpful to achieve fast convergent performance, as proven in [239].

3) Evaluate the fitness

A Kriging model is constructed in this step using the given samples (e.g. initial population/swarm is used in the first iteration) to estimate the fitness and MSE of all updated individuals/particles. The

WMEI introduced in chapter 3 is then applied to guide the selection of which individual or particle should be evaluated using the real objective function. For a better understanding, the WMEI equation is re-defined here as follow

$$E[\boldsymbol{I}(\boldsymbol{x}^*)] = \begin{cases} \boldsymbol{w}[y_{min} - \hat{\boldsymbol{y}}(\boldsymbol{x}^*)]\Phi\left(\frac{y_{min} - \hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right) + (1-\boldsymbol{w})\hat{\boldsymbol{s}}(\boldsymbol{x}^*)\phi\left(\frac{y_{min} - \hat{\boldsymbol{y}}(\boldsymbol{x}^*)}{\hat{\boldsymbol{s}}(\boldsymbol{x}^*)}\right) & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*) > 0 \\ 0 & if\ \hat{\boldsymbol{s}}(\boldsymbol{x}^*) = 0 \end{cases}, \quad (5.5)$$

where $\boldsymbol{w}$ is a $n * 1$ vector of weight factors of individuals or particles. The weight factor may be set to constant, but it is difficult to define a proper value for balancing the exploitation and exploration over the whole optimization process. To adaptively balance the exploitation and exploration, in the proposed algorithms, an adaptive and effective method based on Euclidean distance is proposed to be

$$\boldsymbol{D}_i = \|\boldsymbol{x}_i - \boldsymbol{x}_{min}\| \quad (5.6)$$

$$\boldsymbol{D}_i^{normlized} = \frac{\boldsymbol{D}_i - \overline{\boldsymbol{D}}}{\overline{\boldsymbol{D}} - \underline{\boldsymbol{D}}} \in (0,1) \quad (5.7)$$

$$\boldsymbol{w}(\boldsymbol{x}_i) = \frac{1}{1 + 9 * exp\,(-4.4 * \boldsymbol{D}_i^{normlized})} \in (0.1, 0.9) \quad (5.8)$$

where $\|\cdot\|$ is Euclidean Norm, $\boldsymbol{x}_{min}$ is the position vector of the current minimum, (5.6) measures the distance between each individual and current minimum, $\overline{\boldsymbol{D}}$ and $\underline{\boldsymbol{D}}$ denote the maximum and minimum distances calculated from (5.6), respectively, (5.7) normalizes the measured distances within range from zero to one, and (5.8) calculates the weight factor for each individual within range from 0.1 to 0.9. Note, (5.8) is a modified equation inspired from [177], which was originally used to map the inertial weight of PSO as a sigmoid form. By substituting the calculated $\boldsymbol{w}(\boldsymbol{x}_i)$ into (5.5), it puts more emphases on exploration if the offspring/particle is close to the current minimum, or on exploitation if the offspring/particle is far from the current best. Moreover, the calculated weight factor from (5.8) is being limited between 0.1 and 0.9, because the idea is not to perform purely exploitative (i.e. $\boldsymbol{w}(\boldsymbol{x}_i) = 1$) or explorative (i.e. $\boldsymbol{w}(\boldsymbol{x}_i) = 0$) search. Finally, the offspring/particle that has WMEI value is selected and evaluated using real objective function. By doing so, the proposed Kriging individual-based strategy is able to reduce the number of necessary function evaluations at each generation as well as the computational cost.

This step is time-consuming for any evolutionary based algorithm, because many function evaluations of individuals/particles are needed to be computed. In order to reduce the total number of function evaluations, the proposed Kriging individual-based strategy, same as the individual evolution control method, suggests that only one promising offspring/particle from the entire population/swarm should be selected for function evaluation. In other word, most of other individuals/particles are intuitively insignificant for finding better solution and thus are unnecessary

to be evaluated. This idea is inspired from the EGO technique, but the main difference is that the proposed method only locally searches the WMEI from the population/swarm at each iteration; whereas EGO requires an optimizer to find the global WMEI point from the whole solution space that is a nontrivial optimization task and computationally inefficient. Although the proposed strategy conservatively locates the local WMEI from some given points (i.e. offspring/particles), it is claimed as a more computationally efficient way than EGO because the selections follow the evolutionary trajectory where from the offspring/particles to the global optimum. The EGO seems to be more reliable in finding the global WMEI, but the required computation cost may not be acceptable when solving the OPF problems. Moreover, the iteratively changed WMEI results in a very difficult dynamic optimization problem, and the optimizer used in EGO cannot guarantee to find the global WMEI as well as the global optimum.

4) Updating the population/swarm and samples

A possible strategy of updating the population/swarm, as recommended by the evolution control strategy, is that a certain number of individuals or particles (e.g. selected based on their estimation, MSE or WMEI) use the evaluated fitness, while others use their estimated fitness. As a result, the fitness in the population consists of both estimated and evaluated fitness. However, such strategy may push the optimization process towards a false optimum introduced by the surrogate model and result in unwanted additional function evaluations, should the constructed surrogate model have low fidelity. To prevent this situation, the proposed algorithm suggests that the estimated fitness should not be considered for updating the population/swarm unless one can ensure the surrogate model is high-fidelity. Since only one individual or particle is evaluated using objective function at each generation, the proposed approach of updating the population/swarm is that, the selected infill point that has WMEI value should be used to replace the individual (for KAGA) or personal best (for KAPSO) that has maximum fitness. The Pseudo-code of updating population and swarm may be found in the line 14 of Fig. 5.2 and 5.3, respectively. Even if the fitness of the selected infill point is larger than the fitness of the replaced individual or personal best, the infill point suggested by WMEI should be used as a replacement because of the important information it contains (i.e. a point that balances the local search and global search).

In general, or as in EGO, the infill point needs to be iteratively added into the sample set to build a global Kriging model. However, this might significantly increase the computational cost and require large amounts of physical memory, due to the need of inverting a large correlation matrix. For the sake of maintaining the computational cost to an acceptable level when using Kriging to solve OPF problems, this work suggests that using fixed number of samples (i.e. same size with the population/swarm) to build a local Kriging model. This work proposed a “Better-Adoption and Worse-Abandonment” method used to update the sample set and Kriging model. The Pseudo-code may be found in the line 9-13 of Fig. 5.2 and 5.3, where the infill point replace the minimum sample point only if the fitness of infill point is smaller than or equal to the minimum sample, otherwise, the

infill point will replace the sample that has maximum fitness. By doing so, most of the samples exist in the exploitative region around current minimum and a small number of distinctive samples are in the explorative region that is far from current minimum. This is claimed to be an effective way to alleviate the computing issue of Kriging estimation associating with the sample size, while a local Kriging model around the global optimal solution region could be eventually constructed along with the evolutionary optimization process.

5) Elitist Learning Strategy

In order to avoid the individuals/particles being trapped at the local optimum, it may need an additionally effective mutation operator. A well-performed Elitist Learning Strategy (ELS) proposed in [240] to solve numerical optimization problems is directly employed to enhance the exploration capability of the proposed KAGA and KAPSO algorithms to jump out the local optimum. The simple ELS may be express as follow

$$x_{els} = \boldsymbol{x}_{min}^{d} + \left(\overline{\boldsymbol{x}}^{d} - \underline{\boldsymbol{x}}^{d}\right)N(0,\sigma) \tag{5.9}$$

where d is uniformly random selected so that every dimension has equal probability to be mutated, $\boldsymbol{x}_{min}^{d}$ is the value of current minimum in d^{th} dimension, $\overline{\boldsymbol{x}}^{d}$ and $\underline{\boldsymbol{x}}^{d}$ denote the upper and lower bounds of the variable in the d^{th} dimension, and $N(0,\sigma)$ is a normally distributed random number with mean zero and standard deviation σ. In order to ensure the convergence performance, the σ is generally set to decrease linearly along with the process of iterations from 1 to 0.1. However, it may be (although this tends to be problem dependent) that a constant value results in a better convergence. The formulation of ELS is similar to the Gaussian mutation used in GA [241], but the difference is that only the current best solution is selected to be mutated. Moreover, in this ELS based mutation, only one dimension is mutated (i.e. each variable has very low mutation probability) at a time because some variables of current best solution may actually belong to the global optimum. After evaluating the mutated current best position, the population/swarm and the sample are updated using the same procedure as in step 4. The Pseudo-code may be found in the line 17-22 of Fig. 5.2 and 5.3. On the other hand, in a similar way as when updating the population/swarm, the mutated current best solution will be used even if it possesses a larger than the maximum fitness in the population/swarm, because of its usefulness in global searching.

6) Stop Criterion Check

The proposed algorithm then goes back to step 2 and continues until the predefined maximum number of function evaluations is reached or another termination criterion is met (e.g. the obtained optimal solution remains unchanged for a certain number of iterations or the maximum allowance computation time is reached).

Algorithm 1: KAGA

1. Generate the initial population, initialize the essential parameters, and set $FEs = 0$
2. **While** $FEs <= MaxFEs$
3. Perform **Roulette Wheel Selection** using (5.1);
4. Perform **Single Point Crossover**;
5. Perform **Non-uniform Mutation** using (5.2);
6. Build the **Kriging** to estimate the generated **offspring** using (3.21);
7. Calculate the WEI using (5.5) – (5.8), and find the individual $infill\_x$ which has maximum WEI;
8. Perform **Function Evaluation** on the selected individual $infill\_x$; $FEs = FEs + 1$;
9. **if** $f(infill\_x) <= Y_{min}$
10. $Y_{min} = f(infill\_x)$; $S_{min} = infill\_x$; /*Y_{min} and S_{min} is the fitness and position of minimum sample, respectively*/
11. **else**
12. $Y_{max} = f(infill\_x)$; $S_{max} = infill\_x$; /*Y_{max} and S_{max} is the fitness and position of maximum sample, respectively*/
13. **end if**
14. $ind_{max} = infill\_x$; $f(ind_{max}) = f(infill\_x)$; /*$ind_{max}$ is the position of which individual has maximum fitness*/
15. Perform **Elitist Learning Strategy** on current minimum using (5.9);
16. Perform **Function Evaluation** on the mutated x_{els}; $FEs = FEs + 1$;
17. **if** $f(x_{els}) <= Y_{min}$
18. $Y_{min} = f(infill\_x)$; $S_{min} = infill\_x$;
19. **else**
20. $Y_{max} = f(x_{els})$; $S_{max} = x_{els}$;
21. **end if**
22. $ind_{max} = x_{els}$; $f(ind_{max}) = f(x_{els})$;
24. Update and record the current global best position and fitness;
23. **end While**

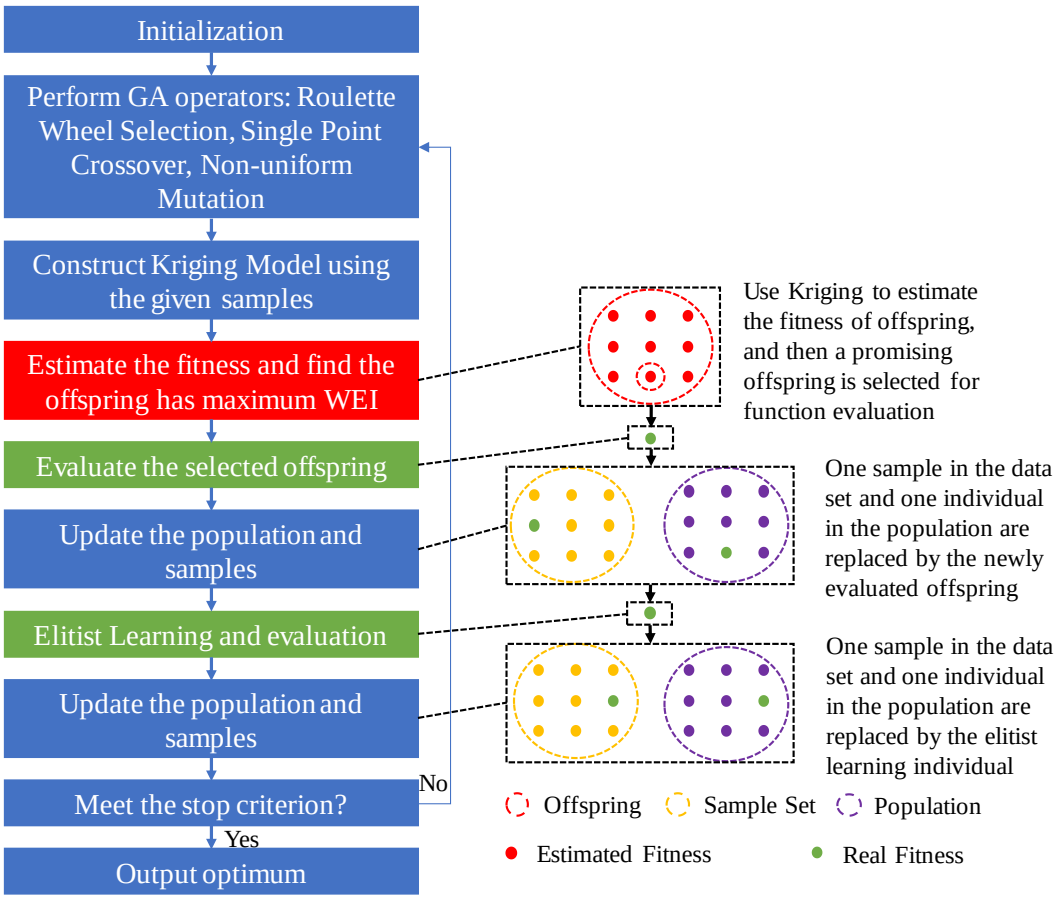


Fig. 5.2 Pseudo-code framework and flow chart of KAGA

Algorithm 2: KAPSO

1. Generate the initial swarm, initialize the essential parameters, and set $FEs = 0$
2. **While** $FEs <= MaxFEs$
3. **for** each swarm particle i **do**
4. Update the velocity v_i and position x_i of particle i using (5.3) and (5.4), respectively;
5. **end for**
6. Build the **Kriging** to estimate the updated particles using (3.21);
7. Calculate the WEI using (5.5) – (5.8), and find the particle $infill\_x$ which has maximum WEI;
8. Perform **Function Evaluation** on the selected particle $infill\_x$; $FEs = FEs + 1$;
9. **if** $f(infill\_x) <= Y_{min}$
10. $Y_{min} = f(infill\_x)$; $S_{min} = infill\_x$; /* Y_{min} and S_{min} is the fitness and position of minimum sample, respectively*/
11. **else**
12. $Y_{max} = f(infill\_x)$; $S_{max} = infill\_x$; /* Y_{max} and S_{max} is the fitness and position of maximum sample, respectively*/
13. **end if**
14. $x_{max} = infill\_x$; $f(x_{max}) = f(infill\_x)$; /* x_{max} is the position of which particle's personal best has maximum fitness*/
15. Perform **Elitist Learning Strategy** on current minimum using (5.9);
16. Perform **Function Evaluation** on the mutated x_{els}; $FEs = FEs + 1$;
17. **if** $f(x_{els}) <= Y_{min}$
18. $Y_{min} = f(x_{els})$; $S_{min} = x_{els}$;
19. **else**
20. $Y_{max} = f(x_{els})$; $S_{max} = x_{els}$;
21. **end if**
22. $x_{max} = x_{els}$; $f(x_{max}) = f(x_{els})$;
23. Update and record the current global best position and fitness;
24. **end While**

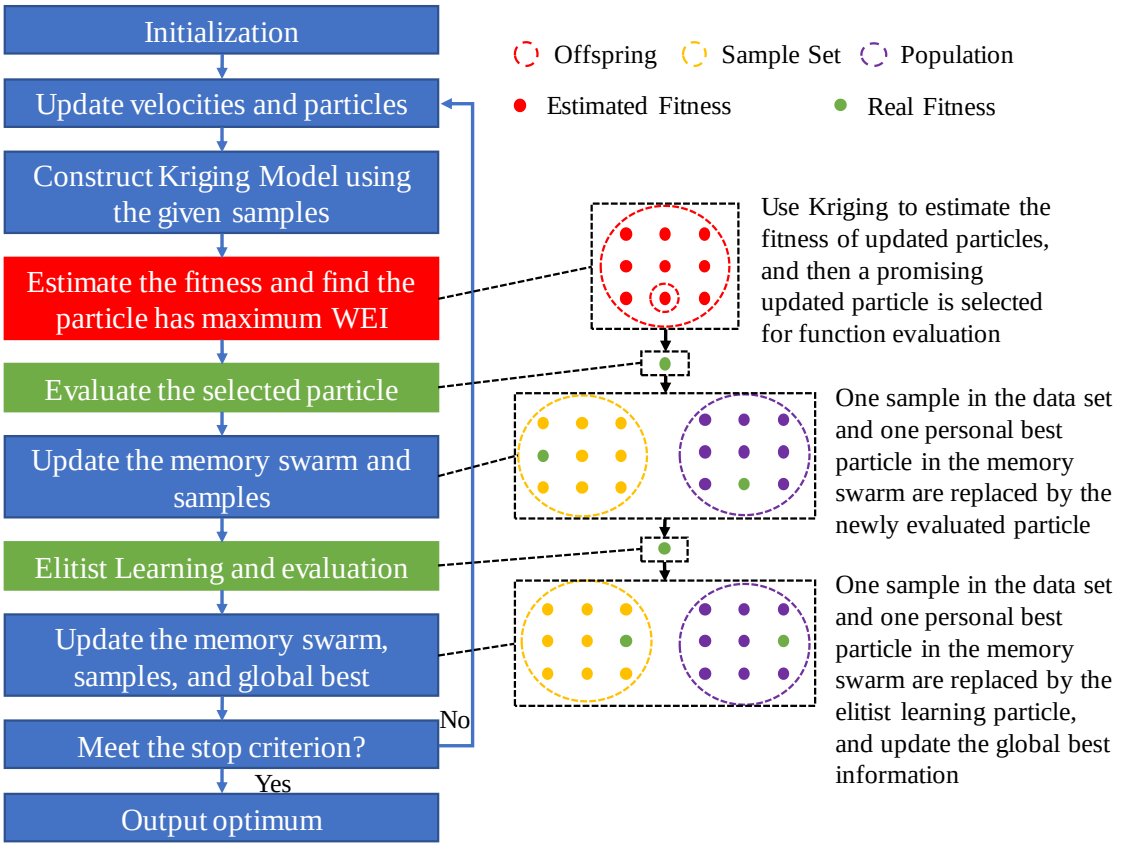


Fig. 5.3 Pseudo-code framework and flow chart of KAPSO

5.3 Numerical Tests on Multimodal Optimization Functions

In order to verify the performance of global optimization, it is not uncommon to test the developed algorithms on the numerical benchmark optimization functions. Although the numerical benchmark functions could not truly reflect the behaviour of OPF problem, it is worthy to use them to demonstrate the proposed algorithms' capability of finding global optimum. Such as [242] applied their proposed algorithm to CEC 2005 benchmark functions to evaluate the efficiency and robustness before solving OPF problem. In this section, the proposed algorithms were tested on five multimodal functions (i.e. Ackley, Griewank, Levy, Rastrigin and Schwefel functions as given in Table. 5.1 and their Matlab codes can be found in the appendix) to demonstrate their convergence performance. Note that all test functions used 50 variables and their accurate global optimums are all equal to zero.

Table 5.1 Multimodal functions

Functions	$f(\boldsymbol{x})$	Range
Ackley	$-a * exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}\boldsymbol{x}_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(c\boldsymbol{x}_i)\right) + a + exp\,(1)$	$-32 \le \boldsymbol{x}_i \le 32$
Griewank	$\sum_{i=1}^{d}\frac{\boldsymbol{x}_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{\boldsymbol{x}_i}{\sqrt{i}}\right) + 1$	$-10 \le \boldsymbol{x}_i \le 10$
Levy	$sin^2(\pi \boldsymbol{w}_i) + \sum_{i=1}^{d-1}(\boldsymbol{w}_i - 1)^2[1 + 10sin^2(\pi \boldsymbol{w}_i + 1)] + (\pi \boldsymbol{w}_d + 1)^2[1 + sin^2(2\pi \boldsymbol{w}_d)]$ $\boldsymbol{w}_i = 1 + \frac{\boldsymbol{x}_i - 1}{4}$, for all $i = 1, \dots, d$	$-10 \le \boldsymbol{x}_i \le 10$
Rastrigin	$10d + \sum_{i=1}^{d}[\boldsymbol{x}_i^2 - 10cos\,(2\pi \boldsymbol{x}_i)]$	$-5.12 \le \boldsymbol{x}_i \le 5.12$
Schwefel	$418.9829d + \sum_{i=1}^{d}\boldsymbol{x}_i sin\left(\sqrt{\boldsymbol{x}_i}\right)$	$-500 \le \boldsymbol{x}_i \le 500$

The original GA and PSO and the proposed KAGA and KAPSO were coded and implemented on Matlab 2018a, as given in the appendix A8 and A9. All these algorithms were run 25 times to test their robustness. The population size and the maximum number of iterations (i.e. *MaxIt*) for these test algorithms were set to 20 and 50,000, respectively. For GA and KAGA, the crossover rate of each selected parents and the mutation rate of each offspring were 0.7 and 0.3, respectively; while for PSO and KAPSO, the constriction coefficients were used. Moreover, the mutation of GA and KAGA is performed by randomly selecting $\lceil 0.01 * nVar \rceil$ variables in the offspring and generating new values, where $ceil$ is the ceiling operator and $nVar$ is the total number of variables. Finally, the standard deviation σ in ELS operator was set to decrease linearly along with the process of iterations from 1 to 0.1 (i.e. $\sigma = 1 - 0.9 * (it/MaxIt)$) to guarantee the convergence, where it is the current counted number of iterations. Note that these parameters were not fine-tuned due to the test functions

could not reflect the OPF problems, and thus they were set empirically to verify whether the proposed strategy could improve the performance of original GA and PSO and help them to locate the global optimum. Nonetheless, some parametric study cases can be found in the Appendix B2-B4.

The results in terms of mean optimal solution ($f(x^*)$), average number of Function Evaluations (FEs) required to find the solution within specified tolerance of 0.01, and the total Success Rate (SR) of finding the tolerance solution for 25 trials obtained from GA, PSO, KAGA and KAPSO are provided in Table. 5.2, where "inf" means that the algorithm failed to converge to the optimum within the specified tolerance after reaching the maximum number of allowed iterations. The results indicate that the GA and PSO are being trapped at local minima for some test functions (e.g. GA for Schwefel function, PSO almost for all test functions), whereas the assistance of Kriging enabled a global or near-global solution to be easily found. Comparing unassisted GA and PSO with the versions utilizing Kriging reveals that using the proposed strategy always improves the performance, in terms of Success Rate, finding better optimum and using smaller number of necessary function evaluations, sometime significantly (e.g. comparing PSO with KAPSO). Finally, comparing KAGA with KAPSO suggests that overall the latter provides higher quality answers (see the bold results). Note that the Kriging assisted approaches consumed more computational time than the GA and PSO in these simple tests, because the test functions are computationally very cheap. However, it has to be emphasized that the more complex the objective function evaluation is, the more efficient the Kriging assisted methods tend to be. According to the test results, it could be concluded that the proposed

Table 5.2 Mean solutions, mean function evaluations and success rate of the 25 trials performed by GA, PSO, KAGA and KAPSO

Algorithms	Index	Ackley	Griewank	Levy	Rastrigin	Schwefel
GA	$f(x^*)$	0.0036	0.0104	2.281E-6	2.074E-4	2.748E+3
	SR	100	44	100	100	0
	FEs	9.504E+5	1.925E+6	1.172E+5	6.763E+5	inf
KAGA	$f(x^*)$	0.0016	**0.0054**	3.5366E-7	4.505E-5	8.632E-4
	SR	100	80	100	100	100
	FEs	6.244E+4	7.124E+4	**1.028E+4**	5.402E+4	6.238E+4
KAPSO	$f(x^*)$	**9.902E-5**	0.0082	**3.476E-11**	**3.415E-7**	**6.364E-4**
	SR	100	80	100	100	100
	FEs	**5.462E+4**	**1.904E+4**	1.296E+4	**5.176E+4**	**2.980E+4**
PSO	$f(x^*)$	3.4188	0.0872	7.9571	81.8253	1.041E+4
	SR	0	24	0	0	0
	FEs	inf	8.565E+4	inf	inf	inf

* Note:
SR = (number of successful runs) / (total runs)
FEs = mean(FEs for successful runs) * (total runs) / (number of successful runs)

Kriging assisted strategy may also be helpful in combination with other optimizations; GA and PSO were chosen here as mentioned is because of their popularity and simple code structure. In next section, the proposed KAGA and KAPSO are tested on the benchmark IEEE test power systems to solve OPF problems.

5.4 Numerical Tests on Continuous Optimal Power Flow Problems

In practice, the power system is operated in a hierarchical look-ahead manner, where the OPF may need to be performed daily in every hour (day-ahead OPF) and hourly in every 5-15 minutes (real-time OPF). By doing so, the power system is economically and securely operated to meet the dynamically changed load consumptions. Currently, the simplified OPF models (e.g. DCOPF) are mainly used in the industry because it could be solved with excellent computation speed by current deterministic solvers. For example, the Economic Dispatch (ED) only optimizes the active power generations without considering other control variables and constraints; the DCOPF considers that all voltage magnitudes are fixed and all voltage angels are zero; the improved decoupled OPF using DCOPF with an AC feasibility check assumes that the active power-voltage angle are independent of reactive power-voltage magnitude (widely used by American power system operators). Although the simplified OPF formulations are well developed and performed, it still urges the need of using ACOPF with increased modelling of system controls and constraints (e.g. include voltage magnitude and reactive power generation) in practical application [87], [243]. By the reason that the ACOPF can model the actually physical behaviour of power system, resulting in precise dispatches and better market signals [87]. Moreover, it has been evidenced that the improvements to ACOPF could largely save the generation costs [8]. In addition, the current deterministic solvers using Newton or Lagrange method to solve the simplified OPF model that is like a process of “approximation of approximation model”, which would eventually provide a sub-optimum or local optimum.

Nevertheless, the nonlinear and highly constrained ACOPF has not been applied in power system operation because of no fast and robust solution technique, whereas the simplified OPF using current solution techniques can be regarded as a trade-off between optimality and computational speed. These concerns have highly motivated this work to develop the algorithm which could solve ACOPF efficiently while maintaining the optimality. On the other hand, the dispute over whether considering the low cost of reactive power controls (e.g. transformer taps, capacitor shunts, voltage magnitudes and reactive power generations) during the dispatch somewhat has limited the application of ACOPF in practice[19], [28]. While considering these reactive power control variables participate by coordinating power losses in the overall cost minimization, although they do not directly affect the generation costs. Therefore, these variables and constraints are considered in this work. Besides, it could make a fair comparison with other reported algorithms as they usually consider reactive power controls. Furthermore, it has to emphasize that the available number of control variables in practical

power system operation is actually limited, especially when the OPF is performed close to the operating time point. Such as, the long-term bilateral contracted generators are adjusted according to their own output profiles; the day-ahead committed generators may not be dispatchable at real-time operation; the real-time dispatchable generators may not participate to AGC; and the transformer taps and capacitor shunts may not be controlled by power system operator, as they could only be regulated locally. Considering those requirements could make the OPF tests more practical, but those factors are in fact system dependency and hard to be defined in the study cases. Without loss of generality, this work fully utilizes the operational capabilities (i.e. all control variables are available) of the test power systems to determinate the global optimality of proposed algorithms, despite the OPF problems would become more complex and difficult. Finally, this work considers "hard" limits for all variables, because it is very difficult to answer that how the "soft" limits are tolerated and to what degree, if the well-defined system operation rules are not available. In the following sections, two ACOPF (later called OPF) problem formulations are introduced and tested.

5.4.1 OPF Formulation

1) Objective Functions

In this chapter, two general objective functions are studied:

- Minimization of generation cost

The minimization of generation cost is one of the most common OPF problems in the literature and it is the one of the main concerns in practice. It aims at achieving economic dispatch during the daily power system operation. The mathematic formulation may be expressed as

$$f_{cost} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{P}_{Gi}^2 + \boldsymbol{b}_i \boldsymbol{P}_{Gi} + \boldsymbol{c}_i, \tag{5.10}$$

where $\boldsymbol{a}_i$, $\boldsymbol{b}_i$, and $\boldsymbol{c}_i$ and $\boldsymbol{P}_{Gi}$ are the cost coefficients and active power output of i^{th} generator, respectively, for calculating the total generation cost. Note that the generators may not use same generation cost function in practice. Above (5.10) is the widely used quadratic cost function in case study cases, while the other cost functions – like piecewise quadratic cost function and quadratic cost curve with valve-point effect – may be found in [91].

- Active power losses

Large amounts of electricity could be saved by minimizing active power losses and it is particularly important when the system under a heavy loaded situation. The objective function could be written as

$$f_{losses} = \sum_{\substack{i=1 \\ k \neq i}}^{n} [\boldsymbol{V}_i^2 - \boldsymbol{V}_k^2 - 2\boldsymbol{V}_i \boldsymbol{V}_k cos\,(\boldsymbol{\delta}_i - \boldsymbol{\delta}_k)]\, \boldsymbol{G}_{ik} \tag{5.11}$$

where $\boldsymbol{G}_{ik}$ is the line conductance between two buses, $\boldsymbol{V}_i$ and the $\boldsymbol{V}_k$ are the voltage magnitudes at the sending bus i and the receiving bus k, while $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_k$ are the voltage angles at the i^{th} and k^{th} bus, respectively.

2) State and control variables

For an OPF problem, there are two types of variables usually considered, which are the state variables $\boldsymbol{x}$ that reflects the system state after operational control and the control variables $\boldsymbol{u}$ that uses to control the system to achieve some operational target (e.g. minimum generation cost). They may be expressed as

$$\boldsymbol{x} = [\boldsymbol{P}_{G1}, \boldsymbol{V}_{d1} \cdots \boldsymbol{V}_{dn}, \boldsymbol{\delta}_2 \cdots \boldsymbol{\delta}_n, \boldsymbol{Q}_{G1} \cdots \boldsymbol{Q}_{Gn}] \tag{5.12}$$

$$\boldsymbol{u} = [\boldsymbol{P}_{G2} \cdots \boldsymbol{P}_{Gn}, \boldsymbol{V}_{G1} \cdots \boldsymbol{V}_{Gn}, \boldsymbol{T}_1 \cdots \boldsymbol{T}_l, \boldsymbol{Q}_{sh1} \cdots \boldsymbol{Q}_{shn}], \tag{5.13}$$

where the control variables $\boldsymbol{u}$ may include active power outputs ($\boldsymbol{P}_{Gn}$) of all generators except the slack bus $\boldsymbol{P}_{G1}$, the voltage magnitudes of all generators ($\boldsymbol{V}_{Gn}$), the transformer taps ratio ($\boldsymbol{T}_l$,), and the switchable shunt capacitor ($\boldsymbol{Q}_{shn}$); the state variables $\boldsymbol{x}$ obtained through power flow calculation consist of the active power output of slack bus generator ($\boldsymbol{P}_{G1}$), voltage magnitudes of all load buses ($\boldsymbol{V}_{dn}$), voltage angles ($\boldsymbol{\delta}_n$) of all buses except slack bus, and the reactive power outputs of each generator, where l and n are the number of transmission lines and buses, respectively.

3) Constraints

For an ACOPF problem, it usually uses following two types of constraint to model the actual physic of power system:

- Equality constraints

The active and reactive power flow balance equations are the most important model to describe the power system behaviour, and they are considered as equality constraints for OPF problems

$$\boldsymbol{P}_i = \boldsymbol{P}_{Gi} - \boldsymbol{P}_{Di} = \sum_{j=1}^{n} \boldsymbol{V}_i \boldsymbol{V}_j \boldsymbol{Y}_{ij}\, cos\big(\boldsymbol{\theta}_{ij} - \boldsymbol{\delta}_i + \boldsymbol{\delta}_j\big) \tag{5.15}$$

$$\boldsymbol{Q}_i = \boldsymbol{Q}_{Gi} - \boldsymbol{Q}_{Di} = -\sum_{j=1}^{n} \boldsymbol{V}_i \boldsymbol{V}_j \boldsymbol{Y}_{ij}\, sin\big(\boldsymbol{\theta}_{ij} - \boldsymbol{\delta}_i + \boldsymbol{\delta}_j\big) \tag{5.14}$$

where $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$ are the injected power at bus i, $\boldsymbol{P}_{Gi}$ and $\boldsymbol{Q}_{Gi}$ are the generator active and reactive power at bus i, $\boldsymbol{P}_{Di}$ and $\boldsymbol{Q}_{Di}$ are the demand active and reactive power at bus i, and $\boldsymbol{\theta}_{ij}$ is the line admittance angle between two buses. Ideally, the injected power at the bus should just equal to the generation power minus the demand power, resulting balanced power flow.

- Inequality constraints

In order to make the system secure, the state variables and control variables should fall within their limits. To achieve this the following inequality constrains are prescribed

$$\boldsymbol{P}_{Gi}^{min} \leq \boldsymbol{P}_{Gi} \leq \boldsymbol{P}_{Gi}^{max} \tag{5.16}$$

$$\boldsymbol{Q}_{Gi}^{min} \leq \boldsymbol{Q}_{Gi} \leq \boldsymbol{Q}_{Gi}^{max} \tag{5.17}$$

$$\boldsymbol{V}_{i}^{min} \leq \boldsymbol{V}_{i} \leq \boldsymbol{V}_{i}^{max} \tag{5.18}$$

$$|\boldsymbol{L}_{l}| \leq \boldsymbol{L}_{l}^{max} \tag{5.19}$$

$$\boldsymbol{T}_{l}^{min} \leq \boldsymbol{T}_{l} \leq \boldsymbol{T}_{l}^{max} \tag{5.20}$$

$$\boldsymbol{Q}_{ci}^{min} \leq \boldsymbol{Q}_{ci} \leq \boldsymbol{Q}_{ci}^{max} \tag{5.21}$$

where the terms in (5.16)-(5.19) used to express active power output limit at slack bust, reactive power output limits, voltage magnitude limits and line flow limits, indicate the degree of reliability and security after the control variables have been adjusted (i.e. the state viable limits). The other constraints specify the acceptable ranges of control variables. Moreover, these variables and limits are assigned by real-valued when applied to the tested optimization cases. Consequently, only the state variable constraints are actually considered when using the proposed algorithms, because the control variables are always enforced within the feasible ranges during the optimization process. Finally, no discrete control variables are considered in this study cases, but it is studied in the next chapter and it can be readily handled by the proposed algorithms using rounding method.

4) Penalty Function

To impose the above state variable constraints, a penalty function is added to the objective function and expressed as

$$f = f + \lambda \tag{5.22}$$

$$\lambda = \boldsymbol{k} \sum_{i}^{n} \boldsymbol{p}(\boldsymbol{x}_i) \tag{5.23}$$

$$\boldsymbol{p}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^{min} - \boldsymbol{x}, & if \quad \boldsymbol{x} < \boldsymbol{x}^{min} \\ \boldsymbol{x} - \boldsymbol{x}^{max}, & if \quad \boldsymbol{x} > \boldsymbol{x}^{max} \\ 0, & if \quad \boldsymbol{x}^{max} \geq \boldsymbol{x} \geq \boldsymbol{x}^{min} \end{cases} \tag{5.24}$$

where $\boldsymbol{k}$ is a self-defined value representing the penalty weight factor, while (5.24) is a fuzzy decision-making function to calculate the penalty function $\boldsymbol{p}(\boldsymbol{x})$. Note that the constraints of control variables can be easily handled by the proposed algorithm, while the penalty function is only used in the proposed algorithms for dealing with the state variables constraints described by (5.12).

5.4.2 OPF Results and Discussion

For the OPF analysis, the proposed KAGA and KAPSO algorithms were implemented by integrating GA/PSO with Kriging, and MATPOWER [244] in Matlab on a PC (Intel Core i7-4790 @ 3.6 GHz, 16 GB memory) for minimizing generation cost (Case 1) and active power losses (Case 2) of the IEEE 30 and 118 bus systems. The related Matlab codes can be found in the appendix A8 and A9. Detailed information about the test systems may be found in MATPOWER casefiles. The upper and lower tap ratings are 1.1 and 0.9 p.u., respectively. Other GA, PSO KAGA and KAGA parameter settings were the same as for the benchmark functions of previous section, except for the population size and the maximum number of iterations which were chosen in accordance with the size of the test problems. Moreover, the number of FEs (i.e. power flow calculations) was selected to be the additional index for comparing the computational performance, rather than only the elapsed time which might be affected by differences in hardware or compilers. Furthermore, the OPF problems were assumed to be continuous, with no discrete variables. In addition, the ELS was applied to both GA and PSO in this study to prevent being trapped in local optima and make more fair comparison. Finally, to improve the convergence speed, the standard deviation in ELS was set to 0.1.

1) IEEE 30 Bus System

The system has 41 transmission lines, 283.4MW active and 126.2MVar reactive power demand, six generators at buses 1, 2, 5, 8, 11 and 13 (with the generator at bus 1 taken as a swing generator), four transformers installed between buses 6-9, 6- 10, 4-12 and 27-28, and nine shunt capacitors at buses 10, 12, 15, 17, 20, 21, 23, 24 and 29. The shunt compensators are adjusted between 0 and 5 MVar. The acceptable voltage magnitudes of generator buses considered can vary between 1.1 and 0.95 p.u., whereas the lower and upper bounds for load buses were set to 1.05 and 0.95 p.u., respectively. In order to ensure that the state variables remain within the limits, the penalty weight factor $\boldsymbol{k}$ described in (5.24) was set to 1000 and 1 for load bus voltage magnitude and reactive power generation constraints respectively. The population size for KAGA and KAPSO needs to be carefully selected, as the larger the population size the better the accuracy of the Kriging model; a large population, however, will increase the computation time because big correlation matrices need inversion. For this small-scale system the population size between 20 and 50 was considered suitable and 50 was actually used. The maximum number of iterations for KAGA and KAPSO was set to 500 (as only two FEs are required at each iteration), a larger value than the one used in GA and PSO (i.e. 100), to ensure the global optimum was found.

The best optimal settings of 50 trials obtained by KAGA and KAPSO for Cases 1 are given in Table 5.3, and the optimal solutions are compared with the results from Efficient Evolutionary Algorithm (EEA) [195], Biogeography Based Optimization (BBO) [245], Moth Swarm Algorithm (MSA) [91], Glow-worm Swarm Optimization (GSO) [246] and Predator-Prey Optimization (PPO) [247] in Table 5.4. The differences between the optimal costs are negligible, but the KAGA and

KAPSO clearly outperformed the other algorithms in terms of shorter computing times. Moreover, when MATPOWER power flow calculation was run using the parameter settings given in literature, the voltage magnitudes of load buses of EEA and BBO were above their specified upper limit (1.05 p.u.), as shown in Fig. 5.4; this implies that KAGA and KAPSO yield more reliable results as all voltages were within the specified limit. Note also that the upper voltage limit of load buses for GSO and PPO was actually set to 1.1 p.u., which is different from other algorithms. In order to make a fair

Table 5.3 Best optimal settings from KAGA and KAPSO

	KAGA		KAPSO			KAGA		KAPSO	
Variables	Case 1	Case2	Case 1	Case 2	Variables	Case 1	Case2	Case 1	Case2
P_{g2}	48.8514	80.0	48.7419	80	T_{6-10}	0.9119	0.937	0.9581	1.028
P_{g5}	21.3771	50.0	21.2718	50	T_{4-12}	0.9883	0.991	0.9698	0.998
P_{g8}	20.7461	35.0	21.1087	35	T_{27-28}	0.9821	0.974	0.9777	0.976
P_{g11}	12.1265	30.0	12.1602	30	Q_{C10}	4.5972	3.795	4.9797	4.421
P_{g13}	12.00	40.0	12.0113	40	Q_{C12}	2.5990	4.849	2.2098	2.377
V_{g1}	1.0797	1.061	1.0825	1.0613	Q_{C15}	5.0	2.679	4.9254	4.844
V_{g2}	1.0621	1.056	1.0641	1.0546	Q_{C17}	3.8280	2.070	4.6838	5.0
V_{g5}	1.0333	1.037	1.0332	1.0344	Q_{C20}	4.3910	2.668	2.9661	3.144
V_{g8}	1.0362	1.043	1.0374	1.0386	Q_{C21}	5.0	5.0	4.9994	5.0
V_{g11}	1.0621	1.090	1.0819	1.0690	Q_{C23}	2.1225	2.382	3.6618	2.986
V_{g13}	1.0544	1.052	1.0398	1.0525	Q_{C27}	5.0	4.971	4.8890	5.0
T_{6-9}	1.0442	1.040	1.0314	0.9837	Q_{C29}	3.3295	2.108	2.8936	3.061

Table 5.4 Comparison of performance (IEEE 30 bus system)

	Case 1		Case 2		
Algorithms	Cost ($/h)	Time (s)	losses (MW)	Time (s)	total FEs
GA	800.51	5.5248	3.0943	6.1664	5,050
KAGA	800.48	1.9863	3.0932	1.9630	1,050
PSO	800.43	5.7212	3.0894	5.4236	50,50
KAPSO	800.45	1.8494	3.0948	1.8907	1,050
EEA	800.08	5.9694	3.2823	5.7167	109
BBO	799.11	11.02	-	-	10,000
MSA	800.51	14.91	3.1005	-	5,000
GSO	799.06	12.26	-	-	6,000
PPO	799.06	10.71	2.867	-	9,000

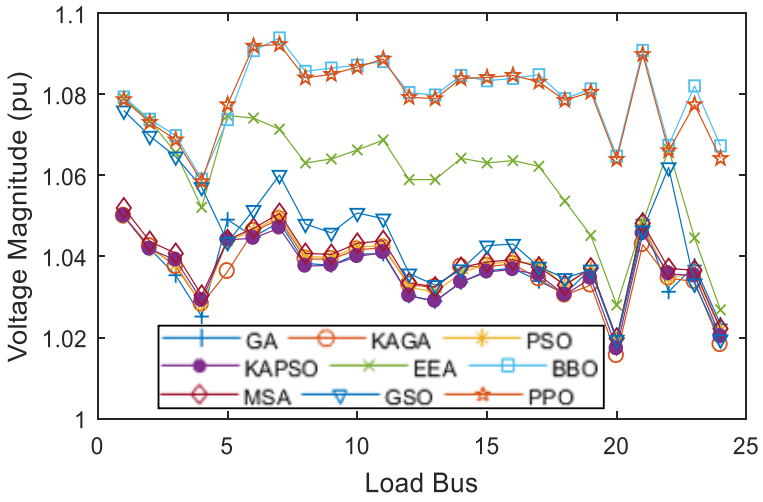


Fig. 5.4 Voltage profiles of load buses for Case 1 (IEEE 30 bus system)

comparison and further verify the proposed algorithms, running KAGA and KAPSO with the limit increased to 1.1 p.u. produced the optimal costs of 798.97 and 798.92 $/h, respectively, with the KAPSO result being overall the best.

For Case 2 of minimizing active power losses, the proposed Kriging assisted approaches performed as good as GA and PSO, while yielding a better optimum than EEA and MSA. Setting the upper voltage limit to 1.1 p.u. (i.e. the same as with PPO) for KAGA and KAPSO resulted in the losses of 2.8389 and 2.8333 MW, respectively; again the KAPSO result was overall the best. Even more importantly, however, KAPSO achieves the result with the shortest elapsed computation time and the smallest number of necessary FEs. Note that EEA appears to require fewer FEs, but it in reality performs many additional sensitivity matrix calculations at each iteration, thus the computation time is significantly longer than for KAPSO.

In the forgoing analysis the maximum number of iterations was used as a stopping criterion; this may be regarded rather conservative. As an alternative – and to assess the robustness of KAGA and KAPSO – a criterion was applied to terminate iterations when the standard deviation of the fitness of the population/personal best becomes smaller than 0.01; this was tested for the case of minimizing the generation cost. The average, best, worst and standard deviation (std) values of the optimal solution, as well as the average CPU time of 50 trials and the number of FEs, are compared in Table 5.5. The GA and PSO required a high number of FEs to converge, but the overall time was still shorter than for other methods reported in literature; this is likely due to the more advanced hardware or more efficient coding, so direct comparisons are somewhat inconclusive. But KAGA and KAPSO have again shown their superior performance being 2.5 and 2.6 times faster than GA and PSO and requiring fewer FEs. In addition, the average CPU times of GA, KAGA, PSO and KAPSO for each iteration were 0.0575s, 0.0039s, 0.0554 and 0.0039s, respectively, thus KAGA and KAPSO were

Table 5.5 Average convergence solutions, function evaluations and CPU time of 50 trials for minimizing generation cost

Algorithms	Mean	Best	Worst	Std	Time(s)	FEs
GA	801.09	800.55	802.37	0.4584	4.8675	3,672
KAGA	801.06	800.58	802.68	0.5037	1.9479	794
PSO	800.54	800.44	800.68	0.0627	3.878	3,366
KAPSO	800.65	800.48	800.89	0.0908	1.485	532
EEA	800.17	800.08	800.21	0.0407	5.969	109
BBO	799.19	799.11	799.20	-	8.816	8,000
GSO	799.06	799.05	799.91	-	4.291	2,100
PPO	799.09	799.05	799.13	-	4.280	3,600

approximately 14 times faster (because only two FEs were required at each iteration). Regarding the solution quality, KAPSO offers a slightly better solution in terms of mean, best, worst and std values than KAGA, while the consistency of the optimal solution obtained by KAPSO is quite acceptable (small standard deviation). It can therefore be concluded that the proposed KAPSO algorithm provides better and more reliable solutions with faster convergence, less CPU time and fewer FEs. Furthermore, acceleration of the GA and PSO may be accomplished by using Kriging to reduce the total number of FEs.

2) IEEE 118 Bus System

This larger-scale system has 4,242 MW active and 1,439 MVar reactive power demand with 130 control variables, including 53 active power outputs and 54 terminal voltages of generators, 9 transformers and 14 shunt compensators. The upper and lower limits for voltage magnitudes of all buses are set to 1.06 p.u. and 0.94 p.u., while the shunt capacitors can compensate a minimum of 0 to a maximum of 30 MVar. solving the OPF problems the population size and maximum number of iterations for the proposed algorithms were set to 100 and 10,000, respectively.

In this particular case, we found that the penalty factor used in IEEE 30 bus system for the load bus voltage and the reactive power generation constraints were no longer adaptive (i.e. the penalties for voltage and reactive power, compared with the objective value, were too small and too large, respectively). As a result, the voltage magnitudes and the reactive power generated were easily pushed beyond the specified limits and trapped at local optima. Therefore, the penalty values of 5,000 and 0.02 for the load bus voltage and the reactive power generation constraints were used instead to deal with the inequality constraint of state variables. Such settings usually lead the algorithms to find good optimal solutions, but the reactive power generation may violate the specified limits because the penalty value is set too small. In order to address such reactive power generation violation, the

Table 5.6 Mean, best, worst and std solutions of 50 trials (IEEE 118 bus test system)

Case 1: Minimization of Generation Cost ($/h)				
Algorithms	Mean	Best	Worst	std
GA	129,686	129,647	129,736	20.5944
KAGA	129,724	129,642	129,796	31.0478
PSO	129,660	129,627	129,714	20.8563
KAPSO	**129,637**	**129,619**	**129,662**	**10.2644**
Case 2: Minimization of Active Power losses (MW)				
Algorithms	Mean	Best	Worst	std
GA	10.5648	10.1769	11.2041	0.2100
KAGA	10.8130	10.3051	11.6889	0.3116
PSO	10.1577	9.9292	10.4943	0.1165
KAPSO	**10.0957**	**9.9086**	**10.3695**	**0.1063**

"Reactive Power Enforcement (RPE)" strategy in MATPOWER was employed. The reactive power generation is enforced to its limit and this "PV" bus is re-defined to "PQ" bus for power flow calculations, if any violation is detected at the maximum iteration. This is a very effective strategy to deal with the reactive power generation constraints and usually requires few power flow calculations when compared with the computational effort of searching for a feasible optimal solution within the solution space.

Moreover, the GA, PSO, KAGA and KAPSO were tested for 50 trials to assess their reliability and consistency. It was noted that there were some inconsistencies in the answers related to the mean, best, worst and std values of the 50 trials, a feature not uncommon when dealing with large problems, using limited computational budget and using different parameter settings. The KAPSO performed overall the best in terms of mean, best, worst and std values for case 1 and case 2, whereas KAGA obtained larger std value than GA for case 1 and KAGA performed as good as GA for case 2, as presented in Table 5.6. But only the best solutions obtained from the 50 trials are selected to compare with reported results, as they do not report the solution from multiple trials. Furthermore, in order to ensure that the algorithms would generate truly random values, a MATLAB pseudo-random number generator '*rng*('*shuffle*')' was used at the beginning of the script.

The best cost and best losses of 50 trial obtained from GA, KAGA, PSO and KAPSO simulations (GA and PSO using population size of 100 and the maximum number of iterations 1,000), compared with MSA, TLBO [70] and GPSO [71] results, are presented in Table. 5.7. It can be observed that the Kriging assisted methods perform as good as (or even better than) the GA and the PSO, and again the proposed approaches outperform the GA, PSO and other algorithms in terms of efficiency of computation, both the necessary FEs and actual simulation times. A more detailed inspection of Table 5.7 reveals that KAPSO offers the lowest cost, although marginally worse losses than GPSO. However, GPSO required a massive number of FEs and is computationally very inefficient. Similarly,

TLBO has been shown to be computationally extremely inefficient (1,885s). Comparing GA and KAGA reveals that the minimum cost and minimum losses are similar, but the KAGA requires less computational effort in terms of CPU time and FEs. Moreover, without going into details, the minimum cost provided by TLBO was achieved while violating some reactive power generation constraints (not negligible) and the MSA's solution could not be verified due to the lack of active power generation value at bus 1. The KAPSO results, on the other hand, are consistently within the imposed limits (see Fig. 5.5 and Fig. 5.6) thus the algorithm appears to be very robust.

Table 5.7 Comparison of performance (IEEE 118 bus system)

	Case 1		Case 2		
Algorithms	Cost ($/h)	CPU Time (s)	Losses (MW)	CPU Time (s)	Total FEs
GA	129,647	211.05	10.177	206.23	100,100
KAGA	129,642	84.853	10.305	84.327	**20,100**
PSO	129,627	175.72	9.929	176.39	100,100
KAPSO	**129,619**	**75.744**	9.909	**75.960**	**20,100**
MSA	129,640	-	-	-	50,000
GPSO	192,627	328	**9.833**	-	768,000
TLBO	129,682	1,885	-	-	-

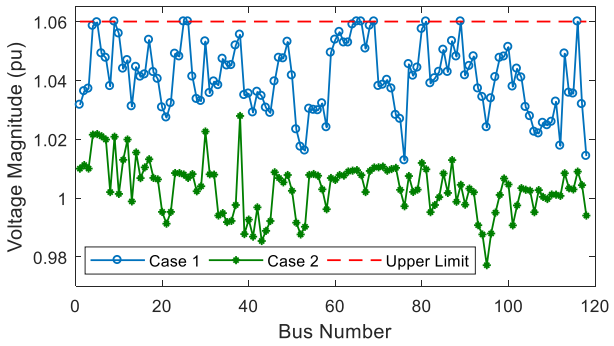


Fig. 5.5 Voltage Profiles of all buses for KAPSO

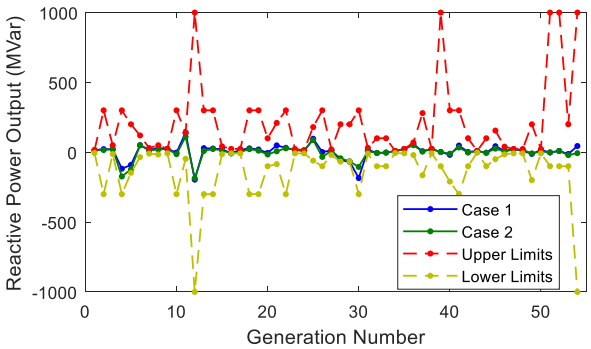


Fig. 5.6 Reactive power generations for KAPSO

5.5 Summary

In this chapter, an individual-based Kriging assisted strategy, inspired from EGO and evolution control, has been proposed to expedite the evolutionary algorithms for the solution of OPF problems. The inherent difficulty of Kriging estimation due to the requirement of matrix inversion – particularly troublesome for large problems – has been alleviated by applying constant sample size. The iterative replacement method is applied to update the Kriging sample as well as the population or memory swarm. To balance exploitation and exploration when selecting a particle with a maximum expected improvement value for Kriging, an efficient Euclidean distance-based method has been proposed. Before applying to power system optimization, two approaches combining Kriging with GA and PSO, called KAGA and KAPSO, respectively, were tested using five complex unconstrained benchmark optimization functions. The results have demonstrated that by providing this assistance by Kriging the total number of necessary function calls has been reduced, sometimes significantly, while the solution quality has been improved. Consequently, KAGA and KAPSO were applied to the IEEE 30 and 118 bus systems to solve two basic OPF problems of minimizing the generation cost and active power losses. The results have shown that the solution quality of the KAGA is as good as the GA, but the computational efficiency of GA in solving OPF problems has been improved by the proposed individual-based Kriging strategy. For both test cases, the KAPSO have demonstrated consistent and robust performance avoiding violation of constraints (associated with other methodologies) and offering good quality solutions with significant improvements in terms of computational efficiency. It is therefore concluded that the proposed approach is very promising for improving the computation efficiency of evolutionary based algorithms when solving OPF problems. Finally, it could be further studied by combining Kriging with other evolutionary based methods and applying the algorithms to other complex OPF problems with larger system sizes.

Chapter 6 Development and implementation of Pre-selection based Kriging Assisted Evolutionary Algorithms

6.1 Introduction

Based on the concept of EGO technique and the evolution control of managing surrogate method in evolutionary computation, the improved individual-based Kriging strategy has been proposed and demonstrated in last chapter. It can potentially prevent the drawback of evolution control caused by the "Curse of Uncertainty" of surrogate model and significantly improve computational performance of canonical version of GA and PSO when solving numerical optimization and OPF problems. However, the proposed individual-based Kriging strategy could be deemed as a greedy technique that benefits from the "Blessing of Uncertainty", because only one WMEI individual/particle selected from the population/swarm is evaluated using real objective function at each iteration. Although the selected WMEI individual/particle is claimed to be the most promising one in the population/swarm and the Euclidean distance-based weight method has been proposed to adaptively balance the exploitation and exploration when selecting an individual/particle for function evaluation, other individuals/particles containing good solutions may be overlooked. Another concern is that the proposed individual-based strategy may degenerate parallel computing nature of evolutionary based algorithms thus it has been limited to series computing environment. For the purpose of addressing the weakness of individual-based strategy, another new pre-selection strategy, which is suitable for parallel computing and able to maximize the benefit of "Blessing of Uncertainty" of Kriging model, is developed in this chapter. The proposed approach is inspired from the ideas of informed operator (or pre-selection) [229], which the production of offspring guided by a surrogate model could speed up the convergence of evolutionary algorithm. The proposed Kriging pre-selection strategy is to locally search around each individual or particle to screen out the most confident one as reproduced offspring or updated particle for real objective function evaluation. In this way, the randomness of particles or individuals generated by operators is intuitively reduced, and thus it is likely to generate better individuals or particles and find better solutions. In other words, the screened individuals or particles would not be worse than carrying them out randomly via evolutionary operator.

The Kriging used to pre-selection may increase the computational complexity of evolutionary algorithm, as the number of function evaluations at each generation or iteration is not reduced while the Kriging estimation is needed before each function evaluation. However, from the convergence point view, the proposed method nevertheless can be greatly helpful in fast locating the precise optimum. Moreover, because the globally high-fidelity surrogate model of large-scale problem is very difficult to construct and computationally inefficiency, the proposed strategy suggests the

candidate points used for Kriging pre-selection that should not be widely spread over the whole solution space. Therefore, the crossover in GA and the velocity update equation in PSO are modified for providing exploitative candidates (i.e. solutions are around the current minimum) to Kriging pre-selection. That is, the current minimum individual is always selected as parent one (other one is selected using roulette wheel selection as in chapter 5) for crossover, and all particles always fly toward the global best position. In this way, it is believed that the Kriging pre-selection is able to maximize the convergence performance while minimizing the risk of wrong selection provided by low-fidelity Kriging model. Clearly, the modified GA/PSO and the Kriging employed to local search somewhat lack exploration capability, thus the ELS used in the previously proposed individual-based algorithms is also applied to the algorithm proposed in this chapter (here called Kriging assisted ELS or KLES). The difference is that the Kriging pre-selection is used to suggest a good mutation value instead of only randomly generating when performing ELS. Besides, a Kriging assisted Orthogonal Experiment Design Learning Strategy (KOEDLS) is proposed to capture the best information or best combination between a local optimum and the current minimum, where the Kriging estimation is used to reduce the required number of function evaluations of factor analysis. The proposed modified GA/PSO operator, KELS and KOEDLS assisted by the proposed Kriging pre-selection could be regarded as local search, global search and balanced search, respectively. In order to properly manage these proposed strategies during the evolutionary optimization process while improving the convergent performance of algorithm, a simple and effective award method is developed. Moreover, the proposed award approach is also used to control the exploitation and the exploration when Kriging pre-selection choosing the best candidate solution for function evaluation. Even though the "Better-Adoption and Worse-Abandonment" strategy for building local Kriging model succeeded in last chapter, it may not be suitable here due to the Kriging pre-selection may need to predict the solution in a wide range of search space. An alternatively effective approach, named "Iteratively Rolling replacement", is proposed to use the newly evaluated solution to replace the oldest sample to build a pseudo global Kriging model. Two algorithms, namely Pre-selection Kriging Assisted Genetic algorithm (PKAGA) and Pre-selection Kriging Assisted Particle Swarm Optimization (PKAPSO), are developed and tested on the numerical functions to prove their performance and effectiveness. Moreover, the individual-based and pre-selection strategy are studied and compared by applying the them to the GA and PSO and their variants. Finally, the developed algorithms are applied to solve the more practical benchmark mixed-integer OPF problems. The obtained solutions are then compared with the ranked algorithms of the benchmark competitions.

6.2 Proposed Pre-selection based Algorithms

The pseudo-codes and the flow charts of PKAGA and PKAPSO are presented in Fig. 6.1-6.4, while the relevant Matlab codes can be found in the appendix A10 and A11. The details of developed PKAGA and PKAPSO algorithms using the proposed strategies are described and discussed below:

1) Initialization

First, the crossover and mutation probabilities of GA and the constriction coefficients of PSO are defined, and the award of each individual/particle, which is used to guide the use of proposed operators and balance the exploitation and exploration, is initialized to zero. A uniformly distributed population/swarm consisting of a user defined number (i.e. p) of individuals/particles is randomly generated and evaluated, while it is used later to build the initial Kriging model. Note, the sample size ns is set to be constant to alleviate the computational cost issue associated with building Kriging model.

2) Kriging pre-selection and individuals or particles update

The reproduction of offspring and update of particle in this step are different from the traditional GA and PSO, because they are always informed by the Kriging pre-selection strategy to suggest the statistically best offspring and particle. Unlike the traditional algorithms, performing the operators once every loop, here the operators in the proposed algorithms are performed a certain number (i.e. nc) of times to generate multiple candidate offspring or particles, which are then estimated by Kriging and the candidate that has WMEI value is selected as the reproduced offspring or updated particle. This Kriging pre-selection, however, may be negatively affected by the low accuracy estimations if the estimation points are widely distributed across the solution space and the limited constant sample size is used to build the Kriging model. In order to mitigate the risk and impact of low-fidelity Kriging built using constant sample size, this work proposes to modify the crossover (for GA) and the PSO to generate the candidates within a small range of solution space, instead of looking for an high-fidelity Kriging model (which is difficult to build especially for high dimensional problems). Even if the Kriging provides low-accuracy estimations, the pre-selection that is made within the candidates generated around a specified point would not be easily influenced by estimation errors, because the Kriging is capable to provide the smooth and continuous surrogate model. In this way, the Kriging pre-selection is employed as a local searcher to refine every local optimum rather than as a global searcher to explore the solution space. Overall, the proposed Kriging pre-selection is deemed to be an enhanced exploitative search process to improve the algorithms' accuracy as well as the convergence performance.

For GA, the roulette wheel selection, single point crossover and non-uniform mutation used in last chapter are also employed here. However, the current best solution is always selected as parent 1, whereas parent 2 is chosen by the roulette wheel selection. The selected parent 1 and parent 2 are

used to perform crossover nc times to produce nc pairs of offspring and the Kriging is then used to screen out the best offspring 1 and offspring 2 according to the WMEI values. The reproduced offspring suggested by Kriging pre-selection could be deemed as the best combinations between selected individual and current best solution. Similar as in the pre-selection assisted crossover, the mutation of selected offspring (i.e. equation (5.2)) is executed nc times to generate nc candidates for Kriging making decision on which candidate value is most likely to find the better solution when it is selected to be the mutation value. Note the selected mutation variable will not change during the process of generating candidate values. Moreover, it can use either different crossover and mutation or different parents and offspring or both within every loop to randomly generate candidate solutions, as recommended in [229]; while this work uses the same selected parents and offspring within every pre-selection assisted crossover and mutation loop to produce candidates that are not being wide spread across solution space. The recommended approaches in [229] could maintain a good diversity of candidate solutions, but they may require higher fidelity degree of surrogate model to select the best candidate.

For PSO, the candidate solutions are also very easy to be generated. First, the modified velocity equation is given as follow:

$$\boldsymbol{v}_i^{k+1} = \omega \boldsymbol{r} \boldsymbol{v}_i^k + c_2 \boldsymbol{r}_2 (\boldsymbol{G}_{best} - \boldsymbol{x}_i^k) \tag{6.1}$$

where i denote the i^{th} particle, k denotes the k^{th} iteration, $\omega = 0.7298$ is the inertia weight factor, $c_2 = 1.4962$ is the global learning coefficient, $\boldsymbol{G}_{best}$ is the vector of global best solution, $\boldsymbol{v}$ is the velocity vector, $\boldsymbol{x}$ is the position vector of particle, and $\boldsymbol{r}$ and $\boldsymbol{r}_2$ vectors have the same length as the particle and are uniformly randomly generated between 0 and 1. This equation without consideration of cognitive (i.e. personal best) term assumes that all the particles in the current swarm should fly toward the current best position to local the precise optimum. The additional random vector $\boldsymbol{r}$ is introduced to increase the diversity of current swarm from dimension to dimension and to reduce the negative impact (e.g. premature convergence) of inertia inherited from local optimum found at previous iteration. To generate the candidate positions of i^{th} particle, (6.1) is performed nc times to give a velocity matrix $\boldsymbol{\mathcal{V}}$ and it then adds with i^{th} particle to provide a position matrix $\boldsymbol{\mathcal{P}}$ (i.e. $\boldsymbol{\mathcal{P}} = \boldsymbol{x}_i^k + \boldsymbol{\mathcal{V}}$). As a result, the velocity matrix $\boldsymbol{\mathcal{V}}$ could provide different paths or speeds for i^{th} particle converging to the optimum, the Kriging pre-selection is regarded as a decision maker to choose the best path or speed. If the s^{th} position in the matrix $\boldsymbol{\mathcal{P}}$ is deemed as the best particle by Kriging, the updated velocity $\boldsymbol{v}_i^{k+1}$ of i^{th} particle is set equal to the s^{th} vector of velocity matrix $\boldsymbol{\mathcal{V}}$, while the updated position $\boldsymbol{x}_i^{k+1}$ of i^{th} particle is set equal to the s^{th} vector of position matrix $\boldsymbol{\mathcal{P}}$. Moreover, the pseudo-codes can be found in line 4-7 of Fig. 6.2.

The Kriging pre-selection strategy is also based on the WMEI criterion, but the difference is that the weight of EI is changed according to the award (the award process is introduced in next step). The proposed Euclidean distance-based weight strategy may no longer be suitable and reliable here,

because the all estimation points are generated around a specified solution point. In order to adaptively balance the exploitation when selecting the promising candidate, the weight of EI used for pre-selection of i^{th} crossover or i^{th} particle update is set to 0 to perform explorative pre-selection if its award is larger than a specified value (i.e. aw_WEI), otherwise it is set to 1 and puts emphasis on exploitation. This means that if i^{th} crossover or i^{th} particle fails to find a better solution than the current best for a specified number of times, its bias of pre-selection will be changed to exploration. Moreover, due to the GA mutation itself being an explorative operator, the weight is fixed to 1 to choose the exploitative offspring, achieving the balance between exploitation and exploration. To this end, with the aid of the proposed Kriging pre-selection strategy, the selection of promising solutions is balanced and the operators are guided/informed, thus the i^{th} pair of reproduced offspring or i^{th} updated particle will not be worse and may be even better than if it is purely randomly generated.

3) Award of individual or particle, evaluation of fitness, and update of sample

The i^{th} pair of offspring or i^{th} updated particle screened-out at the previous step, is evaluated using the real objective function. If the evaluated fitness is better than the current best fitness, the current best record will be replaced by the reproduced offspring or the updated particle. Furthermore, the best offspring reproduced by i^{th} pair of parents will be recorded for managing the use of KOEDLS in later step, while i^{th} particle's personal best will be replaced by its own current particle if the i^{th} particle's fitness is identified as having a better value. Finally, the award process is taking place in this step: if the current best record is updated, the award of i^{th} crossover (i.e. cr_award_i), i^{th} mutation (i.e. mu_award_i) or i^{th} particle (i.e. $award\_i$) will be reset to 0, otherwise its award value will be increased by 1. The award values will be used later to decide whether to perform the KELS and KOEDLS operators.

The "Better-Adoption and Worse-Abandonment" sample update strategy proposed in previous chapter is not employed here, because it builds a local Kriging model and most samples may be gathering around the current best position (may with very small differences), which may lead to overfitting and lack of exploration. It may be suitable for building Kriging at the previous step due to the fact that candidates are generated around the current best. However, the Kriging model in this algorithm is not only used for exploitative pre-selection but also applied to the explorative (i.e. KELS) and the balanced (i.e. KOEDLS) pre-selections. Here, an "Iteratively Rolling Replacement" approach is proposed, where the first sample from the data set is removed and then the newly evaluated individual or particle is added to the sample, set as the $(ns+1)^{th}$ sample, as an update. Nevertheless, because crossover generates two offspring, the sample is updated differently from others, namely the first two samples are removed, and the two offspring are used as $(ns+1)^{th}$ and $(ns+2)^{th}$ samples. The proposed approach here is claimed to be an effective way to build a pseudo-global Kriging model, because the samples are updated following the evolutionary trajectory of individuals or particles

towards the global optimum solution region. For more details, the pseudo-codes of those process can be found in the line 5-12 of Fig. 6.1 and line 8-20 of Fig. 6.2.

4) Kriging Elitist Learning Strategy

Since the crossover and velocity equation are modified to put more emphasis on local search, the GA and especially PSO may be easily trapped in local optimum. The KELS is employed to enhance the algorithm's exploration capability to tackle the premature or local optimum issue. The main difference between KELS and ELS is that the Kriging pre-selection is added to guide how the selected variable of the current best should be mutated, and it helps to achieve better exploration as well as find a better optimum. First, it randomly chooses the d^{th} variable from the current best solution; then (5.9) is used to generate a set of mutation values (i.e. nc values) to provide the candidate mutation solutions, where their value of d^{th} variable is different and the values of other dimensions are all the same; finally, a balanced best mutation is screened based on full exploitation EI (i.e. the weight of EI is set to 1 by the same reason as GA mutation mentioned in previous step), and it is evaluated using the objective function. In addition to using an unchanged d, the d^{th} variable could be randomly selected when generating potential mutations using (5.9), which means the pre-selection could inform which variable could be mutated and how it should be mutated. This might sound like a more robust approach, but it would require the Kriging model to have a higher level of fidelity and thus this potential approach has not been used in this work.

For the purpose of saving on computational cost and to balance the exploitation and exploration during the entire evolutionary optimization process, the KELS operator is only performed when the i^{th} best offspring or i^{th} particle in the previous step failed to find a better optimum; in other words, the award of i^{th} crossover, i^{th} mutation or i^{th} particle (i.e. cr_award_i, mu_award_i or $award\_i$) is larger than 0. The award for triggering KELS could be set to another value (i.e. aw_KELS), but here we set it greedily to zero to secure the global optimum, although this might require more computation effort. Moreover, the award of i^{th} crossover, i^{th} mutation or i^{th} particle is reset to zero if the evaluated fitness of KELS solution is better than the current optimum, otherwise it is left unchanged. Furthermore, the recorded current best solution is updated if the KELS solution is better. Finally, the sample update procedure is the same as in the previous step, thus the newly evaluated KELS solution is added into the sample set as the $(ns+1)^{th}$ sample while the first sample is removed. For more details, the pseudo-codes of those process can be found in the line 13-17 of Fig. 6.1 and line 22-29 of Fig. 6.2.

5) Kriging Orthogonal Experiment Design Learning Strategy

It has been found that the PSO may suffer from "Two Step Forward, One Step Back" phenomenon when particles or individuals exchanging information that some variables are improved but others are deteriorated, as discussed and described in [248]. For example, a 3-D Sphere function $f(\boldsymbol{x}) =$

$\sum_i^d \boldsymbol{x}_i^2$ has optimal position $\boldsymbol{x}^* = [0\ 0\ 0]$, a particle $\boldsymbol{x}1 = [2\ 0\ 9]$ flying toward personal best $\boldsymbol{x}2 = [0\ 3\ 0]$ may become $\hat{\boldsymbol{x}}1 = [1\ 1\ 3]$, which implies that the updated 1st and 3rd variables are closer to the optimum but 2nd variable is changed from a correct position 0 to the poor value 1. This would also happen when two individuals performing crossover, such as $\boldsymbol{x}1 = [2\ 0\ 9]$ and $\boldsymbol{x}2 = [0\ 3\ 0]$ performing single point crossover become $\hat{\boldsymbol{x}}1 = [2\ 0\ 0]$ and $\hat{\boldsymbol{x}}2 = [0\ 3\ 9]$, respectively, where the 3rd variable of $\hat{\boldsymbol{x}}2$ comparing with $\boldsymbol{x}2$ is getting worse. One may argue this would maintain a good exploration capability for algorithm, however it could also slow down and mislead the algorithm. The most obvious way to tackle this issue is to find out all possible combinations between two solutions, but this exhaustion method requires 2^n function evaluations to find out the best combination, where n is the number of total variables. The work [249] proposed that the Orthogonal Experiment Design (OED) technique with the aid of Orthogonal Array (OA) and Factor Analysis (FA) could be used to reduce the number of M combinations from 2^n to $2^{\lceil log_2(n+1) \rceil}$. For example, the $L_4(2^3)$ OA consisting of three factors (i.e. variables) is given as

$$\boldsymbol{L}_4(2^3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}, \tag{6.2}$$

where each row is a possible combination, 1 means that the factor is taken from $\boldsymbol{x}1$, and 2 means that the factor is taken from $\boldsymbol{x}2$. By calculating the fitness of each combination, the best combination among them could be identified using following FA equation

$$\mathbf{S}_{qn} = \frac{\sum_{m=1}^{M} \boldsymbol{f}_m * \boldsymbol{z}_{mnq}}{\sum_{m=1}^{M} \boldsymbol{z}_{mnq}}, \tag{6.3}$$

where $\boldsymbol{f}_m$ is the fitness of m^{th} combination, $\boldsymbol{z}_{mnq}$ is set to 1 if m^{th} combination is analysed the q^{th} level on n^{th} factor, otherwise $\boldsymbol{z}_{mnq}$ is zero. The (6.3) after calculation may be given as

$$\mathbf{S}_{qn} = \begin{bmatrix} 49 & 42.5 & 87.5 \\ 45 & 51.5 & 6.5 \end{bmatrix}, \tag{6.4}$$

where the smaller $\mathbf{S}_{qn}$ value means the q^{th} level on n^{th} factor has better solution quality for a minimization problem. According to the minimum $\mathbf{S}_{qn}$ values at each column, the 2nd level on 1st factor, the 1st level on 2nd factor and the 2nd level on 3rd factor could give a best combination is $[0\ 0\ 0]$.

In order to capture the best variables between local optimum and current best solution, the ODE technique is employed in the algorithm, however, the costly FA still requiring $2^{\lceil log_2(n+1) \rceil}$ function evaluations is not favoured in this work either. Instead, this work proposes to use the Kriging estimation to perform FA and hence the computational cost related to function evaluation is effectively reduced to zero. Note that the computationally efficient OED algorithm proposed in [250] is adopted here. To efficiently manage the use of KOEDLS, and to improve the efficiency of the algorithm, a decision-making approach is proposed. For GA, the KOEDLS is performed when the

following conditions are satisfied: the fitness of the best offspring identified from the i^{th} crossover is better than of its any parent (for mutation, the fitness of i^{th} mutated offspring is better than its fitness before mutation), or the current best solution is updated in steps 3 or 4 and the i^{th} best offspring's fitness is not equal to the current best solution. For PSO, the KOEDLS is performed only if the i^{th} particle's personal best or the currently global best are updated at previous steps, and the i^{th} personal best is not equal to the currently global best. If the above conditions are met, the best offspring found in the i^{th} crossover (for mutation, i^{th} mutated offspring) and the current best solution (for GA) or the i^{th} personal best and the current best solution (for PSO) are used as inputs for OED to build the OA and perform FA to find the best combination between these two vectors. The process of defining the best combination could be viewed as capturing the best variable value for each dimension from a local optimum and the current global optimum. This is a balance between local search and global search, unlike the operators in steps 2 and 4 performing exploitation and exploration, respectively. In order to avoid overlooking the best information caused by Kriging estimation error, this work suggests using real fitness FA and estimated fitness FA alternatively. If the award of the i^{th} crossover, the i^{th} mutated offspring or the i^{th} particle (i.e. cr_award_i, mu_award_i or $award\_i$) is larger than the specified value (i.e. aw_OA), the FA is based on real fitness, otherwise the Kriging FA is used with the purpose of reducing the required number of function evaluations. For high-dimensional optimization, it is recommended to set the award value aw_OA of performing real fitness FA to a larger value to prevent excessive effort and time spent on function evaluations. After defining the best combination, its fitness is evaluated using the real objective function. Furthermore, the best combination will be used to replace the i^{th} best offspring or the i^{th} personal best, if it offers better fitness. On the other hand, if its fitness is overall the best, the current best will be updated by it and the award of the i^{th} crossover, the i^{th} mutation or the i^{th} particle will be reset to zero. Finally, the sample set is updated using a similar way as described in previous steps. For more details, the pseudo-codes of those process can be found in the line 18-29 of Fig. 6.1 and line 29-43 of Fig. 6.2.

6) Update the population or memory swarm

For GA, the fitness of old population and offspring is sorted into an ascending order and then truncating them from $p + 1$ individual to the last individual, which is eliminated the worst individuals and reproduce a new population is according to the "survival of the fittest". For PSO, if the fitness of current particle is smaller than its memorised personal best fitness, this particle will be recorded to update the memory swarm. Moreover, the currently global best information is renewed in this step.

7) Stopping Criterion Check

Go back to step 2 and continues until the termination criterion is met.

Algorithm 3: PKAGA

1. Generate the initial population, initialize the essential parameters

2. **While** *FEs* <= *MaxFEs*

/* The first parent is defined as the current best solution, and this operator selects the second individual denoting as *ind2**/

3. Perform **Roulette Wheel Selection** using (5.1);

/*Kriging choose two offspring from the potential children generated by the selected parents based on the WMEI criterion*/

4. Perform **Kriging Single Point Crossover**;
5. Perform **Function Evaluation** on the *offspring*;
6. **if** $f(of\_best) < f(g\_best)$ /* *of_best* is the best position of two *offspring*, *g_best* is the current global best position*/
7. $f(g\_best) = f(of\_best)$; $g\_best = of\_best$;
8. $cr\_award\_i = 0$; /* *cr_award_i* denote the award of individual *i**/
9. **else**
10. $cr\_award\_i = cr\_award\_i + 1$;
11. **end if**
12. Update the sample set: remove the first two sample, and then use the two *offspring* as the $(n+1)th$ sample and the $(n+2)th$ sample; /* *n* is the total number of samples*/

/*Kriging Elitism Learning Strategy*/

13. **if** *cr_award_i* > *aw_KELS* /*tol1* is the user defined positive integer value*/

/*A best value from a group of random number generated by (5.9) is selected by Kriging for the variable to be mutated*/

14. Perform **KELS**;
15. Update the *g_best* and *cr_award_i*; /*Similar as line 6, 7 and 8*/
16. Update the sample set: remove the first sample and use *x_els* as the $(n+1)th$ sample;

/* *x_els* is the learning individual provided by (5.9) and Kriging*/

17. **end if**

/*Kriging Orthogonal Experiment Design Learning Strategy*/

18. **if** $f(of\_best) < f(ind2)$ *or* $cr\_award\_i = 0$, *and* $f(of\_best) \neq f(g\_best)$
19. **if** *cr_award_i* > *aw_OA* /*tol2* is the user defined positive integer value*/
20. Perform **OED** on *ind2* and *g_best*; /*Using real fitness for OA and FA*/
21. **else**
22. Perform **KOED** on *ind2* and *g_best*; /*Using approximated fitness for OA and FA*/
23. **end if**
24. **if** $f(OA\_best) < f(ind2)$ /*OA_best is the best combination between *ind2* and *g_best given by OED or KOED**/
25. $f(ind2) = f(OA\_best)$; $ind2 = OA\_best$;
26. **end if**
27. Update the *g_best* and *cr_award_i*; /*Similar as line 6, 7 and 8*/
28. Update the sample set: remove the first sample and use *OA_best* as the $(n+1)th$ sample;
29. **end if**

/*Kriging guides the mutation process by suggesting a best mutation value from the potential values generated by (5.2)*/

30. Perform **Kriging Single Point Crossover**;
31. Perform **Function Evaluation** on the *mutation*;
32. Update the *g_best*, *mu_award_i* and sample; /*Similar as line 6 - 12*/
33. Perform **KELS** and update the *g_best*, *mu_award_i* and sample; /*Similar as line 13 - 17*/
34. Perform **KOEDLS** and update the *g_best*, *mu_award_i* and sample; /*Similar as line 18 - 29*/

35. Update the population: sort the *old population* and *offspring* in ascending order, and then select *nPop* non-duplicated best individuals as new *population*; /* *nPop* is the population size*/

36. **end While**

Fig. 6.1 Pseudo-code framework of PKAGA

Algorithm 4: PKAPSO

1. Generate the initial swarm, initialize the essential parameters, and set $FEs = 0$
2. **While** $FEs <= MaxFEs$
3. **for** each swarm particle i **do**
4. Using equation (6.1) to Generate a velocity matrix V;
5. Build a potential particle matrix $\boldsymbol{P}$: $\boldsymbol{P} = x_i + \boldsymbol{V}$; /* x_i is the i^{th} particle in the swarm*/
6. Find the particle in $\boldsymbol{P}$ of which has MWEI using Kriging, and the index number of this Particle in the matrix $\boldsymbol{P}$ is recorded as idx;
7. Update the velocity v_i and particle x_i: $v_i = \boldsymbol{V(idx)}$ and $x_i = \boldsymbol{P(idx)}$;
8. Perform **Function Evaluation** on x_i;
9. **if** $f(x_i) < f(pb_i)$ /* pb_i is personal best of particle i */
10. $f(pb_i) = f(x_i)$; $pb_i = x_i$;
11. $flag_i = 1$; /*flag of particle i */
12. **if** $f(pb_i) < f(g_{best})$ /* g_{best} is the current global best position */
13. $f(g_{best}) = f(pb_i)$; $g_{best} = pb_i$;
14. $award\_i = 0$; /* $award\_i$ is the award of particle i */
15. **else**
16. $award\_i = award\_i + 1$;
17. **end if**
18. **else**
19. $flag_i = 0$;
20. **end if**
21. Update the sample set: remove the first sample and use x_i as the $(n+1)th$ sample;

/*Kriging Elitism Learning Strategy*/

22. **if** $award\_i > aw\_KELS$ /*$tol1$ is the user defined positive integer value*/
/*A best value from a group of random number generated by (5.9) is selected by Kriging for the variable to be changed*/
23. Perform **KELS**;
24. **if** $f(x_{els}) < f(g_{best})$ /* x_{els} is the learning particle provided by (5.9) and Kriging*/
25. $f(g_{best}) = f(x_{els})$; $g_{best} = x_{els}$; $award\_i = 0$;
26. **end if**
27. Update the sample set: remove the first sample and use x_{els} as the $(n+1)th$ sample;
28. **end if**

/*Kriging Orthogonal Experiment Design Learning Strategy*/

29. **if** $flag_i = 1$ *or* $award\_i = 0$, *and* $f(pb_i)) \neq f(g_{best})$
30. **if** $award\_i > aw\_OA$ /*$tol2$ is the user defined positive integer value*/
31. Perform **OED** on pb_i and g_{best}; /*Using real fitness for OA and FA*/
32. **else**
33. Perform **KOED** on pb_i and g_{best}; /*Using approximated fitness for OA and FA*/
34. **end if**
35. **if** $f(OA\_best) < f(pb_i)$ /*OA_best is the best combination between $ind2$ and g_best given by OED or KOED*/
36. $f(pb_i) = f(OA\_best)$; $pb_i = OA\_best$;
37. **if** $f(pb_i) < f(g_{best})$
38. $f(g_{best}) = f(pb_i)$; $g_{best} = pb_i$; $award\_i = 0$;
39. **end if**
40. **end if**
41. Update the sample set: remove the first sample and use OA_best as the $(n+1)th$ sample;
42. **end if**
43. **end for**
44. **end While**

Fig. 6.2 Pseudo-code framework of PKAPSO

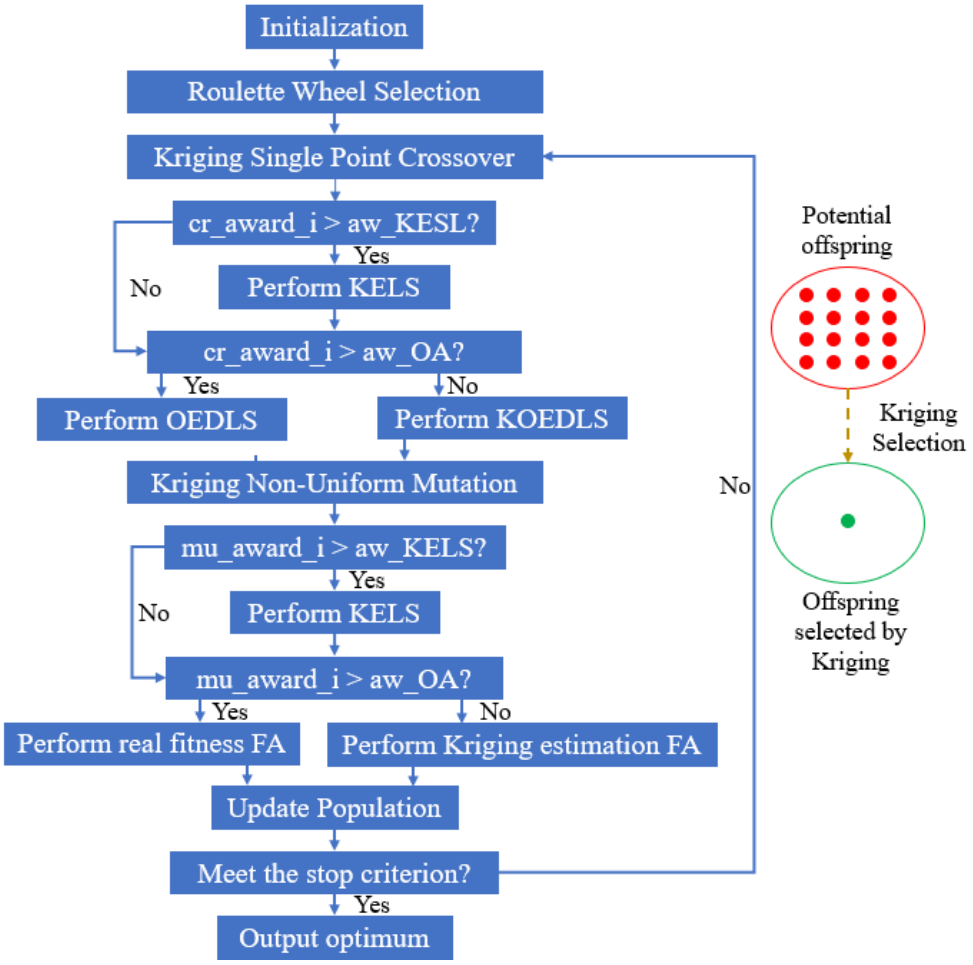


Fig. 6.3 Flow chart of PKAGA

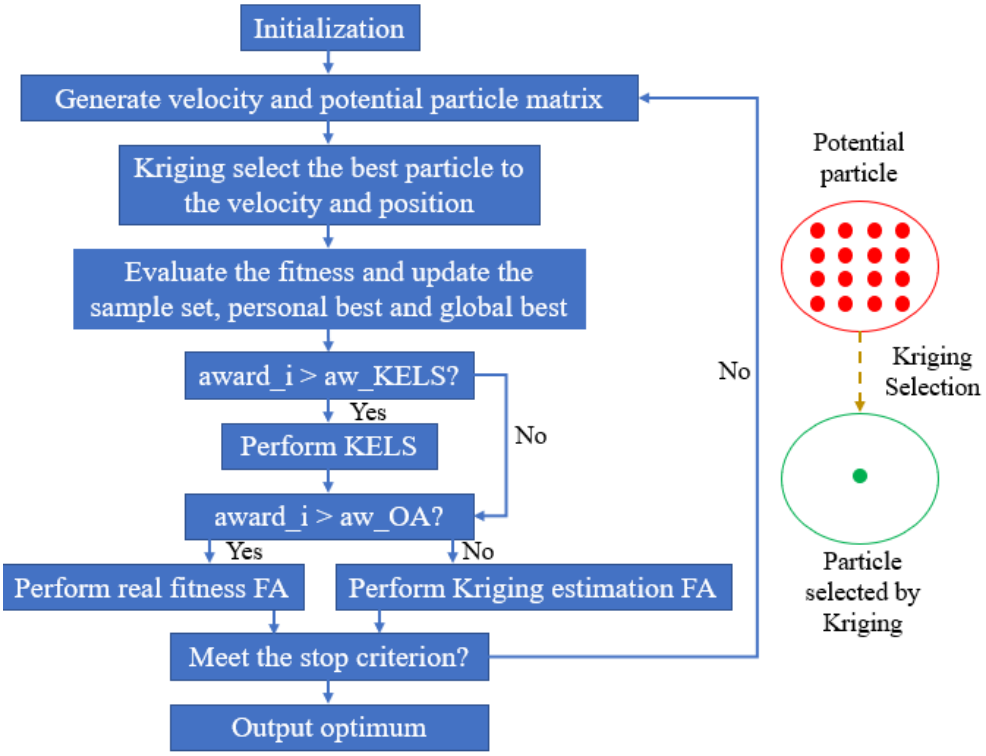


Fig. 6.4 Flow chart of PKAPSO

6.3 Numerical Tests on Benchmark Optimization test Functions

In order to verify the effectiveness of the pre-selection strategy and compare with the previous developed individual-based strategy, both Kriging strategies were not only applied to canonical GA and PSO, but also applied to their well-known variants. For GA, it is not unusual that using different crossover and mutation operators to improve the computational performance, thus the Kriging strategies were also applied to GA that utilizes arithmetic crossover [238] and Gaussian mutation [235]. To improve the performance of PSO, efforts could be made in two aspects: parameter adaptation and population topology adaptation. The representatives of these aspects found from the literature may be: Adaptive Particle Swarm Optimization (APSO) [177] and Comprehensive Learning Particle Swarm Optimization (CLPSO) [251]. Hence the proposed Kriging strategies were also applied to the APSO and the CLPSO. Note, there are plenty of GA and PSO variants can be found in the literature, the selected variants here are only used to illustrate and verify the effectiveness of the proposed Kriging strategies. The details about these variants are not introduced and discussed in this chapter, but one may them find from the referred references. The algorithms were tested on 5 benchmark optimization functions, including the 4 multimodal functions used in chapter 5 (i.e. the relatively easy Levy function was not considered) and 1 well-known unimodal function (i.e. Sphere function). The multimodal functions could be used to study the algorithm's capability in finding global optimum, whereas the unimodal function could be used to check the algorithms' capability in finding accurate optimal solution.

All test function used 50 variables and their accurate optimal solutions are all equal to zero. The tested algorithms were coded and implemented on Matlab 2018a and run 25 times to test their robustness, while the Matlab codes of PKAGA and PKAPSO can be found in appendix A10 and A11, respectively. The population size and the maximum number of function evaluations (i.e. $MaxFEs$) for these test algorithms were set to 30 and 300,000, respectively. Note that the number of candidate solutions (i.e. nc), used for pre-selection, was set equal to the population size (i.e. 30). For those GA related algorithms, the crossover and mutation rates were set to 0.7 and 0.3, respectively, to maintain a good balance between exploitation and exploration. The crossover point of single point crossover was set to be uniformly random selected, the α value used in arithmetic crossover was set to 0.5, and only one variable was randomly selected at a time to be mutated when performing non-uniform or Gaussian mutation. The parameter settings of APSO and CLPSO related algorithms used for this test were set to be the same as stated in their cited references. For the pre-selection Kriging assisted algorithms, the award used to assign the weight of EI when selecting best candidate offspring or particle was set 5 (i.e. $aw\_WEI = 5$), the award of trigging KELS operator was set to 0 (i.e. $aw\_KELS = 0$), and the award of using real fitness FA was set to 0 (i.e. $aw\_OA = 0$). The standard deviation σ used in ELS (for individual based algorithms) and KELS was set to decrease linearly along with the number of function evaluations from 1 to 0.1 (i.e. $\sigma = 1 - 0.9 * FEs/MaxFEs$) to guarantee the convergence, where FEs is the currently counted number of function evaluations.

Table 6.1 Mean solutions and standard deviations of the 25 trials performed by individual-based and pre-selection Kriging assisted algorithms

Algorithms	Versions	Sphere	Ackley	Griewank	Rastrigin	Schwefel
GA	Original	7.21E-3 ± 1.9E-3	1.83E-2 ± 2.9E-3	2.52E-2 ± 2.5E-2	3.33E-3 ± 9.2E-4	2.49E3 ± 4.85E2
	Individual-based	1.50E-5 ± 6.2E-6	7.65E-4 ± 1.9E-4	**6.52E-3 ± 7.9E-3**	8.82E-6 ± 4.4E-6	6.95E-4 ± 3.2E-5
	Pre-selection	**1.25E-6 ± 3.5E-7**	**2.48E-4 ± 3.7E-5**	3.42E-2 ± 4.1E-2	**7.50E-7 ± 2.3E-7**	**6.41E-4 ± 1.7E-6**
GAA	Original	4.39E-4 ± 2.7E-4	4.23E-3 ± 2.1E-3	5.20E-2 ± 4.8E-2	1.73E-4 ± 1.3E-4	2.80E3 ± 4.49E2
	Individual-based	2.62E-6 ± 2.0E-6	3.72E-4 ± 1.4E-4	5.25E-2 ± 5.1E-2	1.17E-6 ± 9.0E-7	6.56E-4 ± 3.7E-5
	Pre-selection	**7.5E-11 ± 2E-10**	**5.56E-5 ± 2.6E-5**	**1.76E-2 ± 2.6E-2**	**8.97E-8 ± 6.5E-8**	**6.37E-4 ± 4.2E-7**
GAG	Original	0.5681 ± 0.2002	0.1897 ± 0.0330	0.5001 ± 0.1159	0.3018 ± 0.0989	1.7663 ± 0.6347
	Individual-based	8.74E-4 ± 2.7E-4	5.02E-3 ± 8.2E-4	**7.21E-3 ± 9.7E-3**	4.49E-4 ± 1.5E-4	2.93E-3 ± 8.9E-4
	Pre-selection	**4.02E-5 ± 9.6E-6**	**1.33E-3 ± 2.2E-4**	3.45E-2 ± 3.1E-2	**2.58E-5 ± 8.0E-6**	**7.69E-4 ± 3.2E-5**
PSO	Original	3.7E-78 ± 1E-77	2.4821 ± 0.5964	5.08E-2 ± 6.8E-2	7.46E1 ± 1.51E1	1.06E4 ± 1.49E3
	Individual-based	9.0E-14 ± 4E-14	1.56E-7 ± 5.1E-8	1.03E-2 ± 1.1E-2	5.9E-13 ± 1E-12	**6.46E-4 ± 6E-12**
	Pre-selection	**0 ± 0**	**2.9E-14 ± 4E-15**	**8.35E-3 ± 1.5E-2**	**8.3E-15 ± 5E-15**	**6.36E-4 ± 2E-12**
APSO	Original	1.6E-88 ± 4E-88	4.01E-7 ± 2.0E-6	2.51E-2 ± 3.9E-2	1.2344 ± 1.2627	4.36E3 ± 2.55E3
	Individual-based	5.61E-6 ± 3.2E-6	3.78E-4 ± 1.0E-4	**1.87E-3 ± 4.4E-3**	6.47E-6 ± 5.1E-6	6.72E-4 ± 3.3E-5
	Pre-selection	**0 ± 0**	**8.6E-13 ± 4E-12**	8.32E-3 ± 1.9E-2	**9.0E-15 ± 5E-15**	**6.36E-4 ± 5E-12**
CLPSO	Original	3.3E-3 ± 9.26E-4	2.84E-2 ± 6.6E-3	7.87E-3 ± 2.5E-3	3.55E1 ± 8.0409	9.48E1 ± 8.37E1
	Individual-based	0.1234 ± 0.0367	6.97E-2 ± 1.4E-2	0.1181 ± 0.0332	4.69E-2 ± 4.5E-2	0.3491 ± 0.0829
	Pre-selection	**2.5E-71 ± 1E-70**	**1.23E-4 ± 3.4E-4**	**9.43E-3 ± 1.7E-2**	**4.3E-15 ± 8E-15**	**6.36E-4 ± 3E-11**

* Notation:
GAA is the Genetic Algorithm using Arithmetic Crossover and Non-uniform Mutation
GAG is the Genetic Algorithm using Single Point Crossover and Gaussian Mutation

The mean and standard deviation solutions of the 25 trials performed on benchmark functions by different algorithms are given in in Table 6.1. It can be observed that the solution quality of original algorithms could be always improved by using the proposed pre-selection strategy (as highlighted by the bolded values), sometime significantly; while the individual-based strategy could be helpful in finding better optimum for most cases. Comparing the pre-selection Kriging assisted algorithms with the individual-based algorithms shows that the algorithms assisted by pre-selection provide better solutions (except the GA, GAG and APSO tested on Griewank function). This indicates that the pre-selection is a more reliable and robust strategy than the individual-based strategy, and it proves the design idea of the pre-selection strategy that the Kriging used in local search phase could find more accurate solution. The PSO assisted by the proposed Kriging pre-selection (i.e. PKAPSO) provided overall better solutions than other algorithms, with the exception of finding solutions for Griewank and Rastrigin functions were slightly larger minima than the individual-based APSO and the pre-selection CLPSO were noted. Moreover, an interesting result is that the pre-selection applied to PSO related algorithms has always outperformed, in the experiments, the pre-selection assisted GA related algorithms. This may suggest that the pre-selection strategy is more suitable for PSO related algorithms. It was not found as a conclusive result by comparing individual-based GAs and

Table 6.2 Mean function evaluations and success rate of the 25 trials performed by individual-based and pre-selection Kriging assisted algorithms

Algorithms	Versions	Sphere (SR / FEs)	Ackley (SR / FEs)	Griewank (SR / FEs)	Rastrigin (SR / FEs)	Schwefel (SR / FEs)
GA	Original	0 / inf	0 / inf	32 / 9.3530E+5	100 / 2.9427E+5	0 / inf
	Individual-based	0 / inf	100 / 1.1259E+5	**68 / 1.4536E+5**	100 / 7.6511E+4	100 / 1.1128E+5
	Pre-selection	**28 / 1.0693E+6**	**100 / 4.7839E+4**	32 / 2.1963E+5	**100 / 2.8640E+4**	**100 / 5.9361E+4**
GAA	Original	0 / inf	100 / 2.6498E+5	24 / 1.0173E+6	100 / 1.8042E+5	0 / inf
	Individual-based	28 / 1.0523E+6	100 / 6.8948E+4	40 / 2.5304E+5	100 / 6.3382E+4	100 / 7.0989E+4
	Pre-selection	**100 / 1.3152E+5**	**100 / 2.2581E+4**	**60 / 4.7299E+4**	**100 / 2.6737E+4**	**100 / 5.6278E+4**
GAG	Original	0 / inf	0 / inf	0 / inf	0 / inf	0 / inf
	Individual-based	0 / inf	100 / 1.7197E+5	68 / 1.6783E+5	100 / 8.7098E+4	100 / 1.6466E+5
	Pre-selection	0 / inf	**100 / 5.4757E+4**	**28 / 1.4391E+5**	**100 / 4.2396E+4**	**100 / 6.9040E+4**
PSO	Original	100 / 3.3200E+4	0 / inf	36 / 5.0097E+4	0 / inf	0 / inf
	Individual-based	100 / 5.0722E+4	100 / 6.5628E+4	60 / 2.6382E+4	**100 / 6.4172E+4**	**100 / 2.6768E+4**
	Pre-selection	**100 / 8.6565E+3**	**100 / 1.3428E+4**	**72 / 1.2869 E+4**	100 / 8.5477E+4	100 / 7.2748E+4
APSO	Original	100 / 3.1680E+5	100 / 7.4212E+4	60 / 3.3312E+4	36 / 7.7427E+5	0 / inf
	Individual-based	0 / inf	100 / 8.7882E+5	88 / 6.9854E+4	**100 / 8.0195E+4**	100 / 1.0049E+5
	Pre-selection	**100 / 8.3839E+3**	**100 / 1.1517E+4**	**76 / 1.4084E+4**	100 / 8.2467E+4	**100 / 8.7114E+4**
CLPSO	Original	0 / inf	0 / inf	**76 /** 3.8370E+5	0 / inf	0 / inf
	Individual-based	0 / inf	0 / inf	0 / inf	0 / inf	0 / inf
	Pre-selection	**100 / 4.4224E+4**	**100 / 7.4848E+4**	68 / **8.0521E+4**	**100 / 4.8475E+4**	**100 / 4.3182E+4**

* Notation:
SR = (number of successful runs) / (total runs)
FEs = mean(FEs for successful runs)
inf = failed to find the acceptable optimum

individual-based PSOs, but it was noticed that the individual-based PSO, proposed in previous chapter [31], performed better than other individual-based algorithms.

In addition to the algorithms' solution quality, the average number of Function Evaluations (FEs) needed to find the solution within specified accuracy level (i.e. 0.01 for Sphere and 1E-6 for others) and the total Success Rate (SR) of finding the acceptable optimal solution for 25 runs are given in Table 6.2, to demonstrate the reliability and efficiency. Note that the CPU time of Kriging assisted algorithms (especially individual-based algorithms) were actually much longer than the original algorithms, due to the fact that the benchmark function evaluation is computationally very cheap. For those time-consuming optimization problems, the reduced necessary FEs means significant reduction in computational time saving. According to the results in terms of FEs and SR, it is difficulty to conclude which algorithm is the best, because all the tested algorithms have good performance in some of the tested functions and less so in others. Nonetheless, the pre-selection assisted algorithms (especially the PSO related algorithms) required less FEs than their original versions and had a higher success rate for almost all cases. Also, it would not degenerate the performance of original algorithms in the worst scenarios. For the individual-based strategy, it indeed

improved the performance of some original algorithms, but it also degenerated some algorithms, for example the required FEs of individual-based APSO for Ackley function was almost 12 times more than its original version, and the success rate of CLPSO in Griewank function was 76% whereas the individual-based CLPSO was 0. Therefore, the overall results recommend that the pre-selection is a more advance, accurate, reliable and efficient Kriging strategy to improve the performance of evolutionary based algorithms. Finally, comparing the developed algorithms suggests that overall the PKAPSO outperforms PKAGA, in terms of higher success rate and providing better optimum using smaller number of function evaluations. In order to further verify the computational performance, the proposed strategies and developed algorithms are applied to solve the more practical problems – mixed-integer OPF – in next section. Furthermore, the comparisons between individual-based strategy and pre-selection strategy can be found in appendix B5-B7, as well as the study of candidate solution size associated with PKAGA and PKAPSO.

6.4 Numerical Tests on Benchmark Mixed-integer Optimal Power Flow Problems

The OPF problem considering only continuous variables, like in chapter 5, may not properly model the power system in a practical engineering sense. For example, the transformer On-Load Tap Changer (OLTC) in practice is operated in the way of selecting the discrete steps to change the turn ratios and the shunt compensation may only offer the binary on-off option to the operator. In order to evaluate the applicability and effectiveness of the developed PKAGA and PKAPSO in practice, the power system optimization test bed developed by IEEE Working Group on Modern Heuristic Optimization [252] is used. The OPF test bed is an encrypted file based on the functionalities of Matlab and MATPOWER toolbox in order to perform the evaluation of OPF objective function and constraints. In this way, the black-box OPF problem could be solved by any participated evolutionary algorithm. This OPF test bed includes two test cases – Optimal Active and Reactive Power Dispatch (OARPD) and Optima Reactive Power Dispatch (ORPD) – on three modified versions of IEEE benchmark test systems (57, 118 and 300) and one Offshore Wind Power Plant (OWPP). Some descriptions of the test systems may be found in Table 6.3 and more details can be found in the test

Table 6.3 Composition of IEEE test systems

Components		IEEE 57	IEEE 118	IEEE 300	OWPP
Generators		7	54	69	18
Loads		42	99	201	1
Transmission Lines		63	177	304	20
Transformers	OLTC	15	9	62	2
	Fixed tap	2	0	45	18
Shunt Compensation	Binary on/off	3	14	14	0
	Stepwise	0	0	0	1
	Continuous	0	0	0	1

bed casefiles. Note that the objective function and constraint of OARPD and ORPD are the same as in the case 1 (i.e. minimization of generation cost) and case 2 (i.e. minimization of active power losses) in chapter 5, respectively, but this OPF test bed considers discrete and binary variables (i.e. mixed-integer OPF). Moreover, the active power generations here are no longer used as the control variable to minimize the active power losses (i.e. ORPD), because they would against the principles of economic dispatch. It has been stated that some of test cases are very hard to be solved, due to the challenge of finding feasible solution (i.e. all variables are within specified range).

The fitness value internally provided by the test bed is calculated as follow

$$f = ob + p\sum_{i}^{c} \mathrm{g}_i^2 \tag{6.5}$$

where ob is the calculated the objective function value, f is the calculated fitness, c is the total number of constraints, g is the penalty value of inequality constraint that is calculated in a similar way as in (5.24), and the penalty weight p is internally defined to 10^7. However, the penalty function is not restricted to (6.5), it is allowed and encouraged to use the self-defined approach. In this work, the sum of g is used instead of sum square of g and the penalty weight p used for imposing constraints is set to smaller value according to the previous OPF studies, as giving in Table 6.4. There are two main reason of using such penalty weight values: first using a large penalty value used for fitness calculation would make the algorithms too conservative, that is emphasis on finding the feasible solution rather than the optimal solution, and vice versa; second the large penalty would increase the estimation errors because, the estimation from point to point in the Kriging model could be dramatically changed. Therefore, this work used the proposed "greedy" penalty approach to deal with the inequality constraints and to find the global optimal solution. The modified penalty constraint function is given in the appendix A10. Note that the fitness values presented in this chapter are the outputs of OPF test bed that are calculated based on (6.5).

Table 6.4 Parameter Settings of PKAGA and PKAPSO

Parameters	IEEE 57	IEEE 118	IEEE 300	OWPP
Population size/sample size	30	100	150	20
Number of candidate solutions	20	30	50	10
σ std value used in KELS	0.1	0.1	0.1	0.1
aw_WEI	5	5	5	5
aw_KELS	0	0	0	0
aw_OA	10	20	30	5
$MaxFEs$	50,000	100,000	300,000	10,000
Number of trials	31	31	31	31
p for voltage constraints (OARPD)	1,000	1,000	1,000	-
p for other constraints (OARPD)	1	1	1	-
p for voltage constraints (ORPD)	1,000	1,000	1,000	1,000
p for other constraints (ORPD)	0.01	0.1	1	1

In addition, the proposed algorithms are modified to deal with the discreate and binary variables. As shown in Fig. 6.5, the KELS used for continuous variables mutation is unchanged, but no candidate values are generated for Kriging to select the best candidate. Furthermore, the parameters settings of different test systems – such as population and sample size, number of candidate solutions, maximum allowance of function evaluations (i.e. $MaxFEs$), the award used to assign the weight of EI when selecting best candidate offspring or particle (i.e. aw_WEI), the award of trigging KELS operator (i.e. aw_KELS), and the award of using real fitness FA (i.e. aw_OA) – are given in the Table 6.4. In following sections, the simulation results obtained from the developed PKAGA and PKAPSO are compared with the results given by the ranked algorithms of the test bed competition. Finally, the pre-selection strategy was applied to these ranked algorithms to verify the improvement on convergence performance.

Improved Kriging Elitism Learning Strategy

1. Generate a random integer number m; /*to decide which variable in the current best solution will be changed*/
2. **if** m^{th} variable is continuous variable
3. Use *gbest* to generate a ***gb*** matrix containing s row of *gbest*; /* *gbest* is the current best solution vector and s is the user defined value to control generate how many potential solutions for Kriging*/
4. Generate s number of floating values using (5.9);
5. Perform **Kriging** to select the best value *bv*;
6. **else if** m^{th} variable is discrete variable
7. **if** round(*gbest_m*) = *gbest_m_max* /**gbest_m_max* is the m^{th} variable's upper limit */
8. *bv* = *gbest_m* – 1;
9. **else if** round(*gbest_m*) = *gbest_m_min* /**gbest_m_min* is the m^{th} variable's lower limit */
10. *bv* = *gbest_m* + 1;
11. **else if** *gbest_m_min* < round(*gbest_m*) < *gbest_m_max*
12. **if** *rand* > 0.5 /* *rand* is a random value between 0 and 1*/
13. *bv* = *gbest_m* + 1;
14. **else**
15. *bv* = *gbest_m* - 1;
16. **end if**
17. **end if**
18. **else if** m^{th} variable is binary variable
19. **if** round(*gbest_m*) = 0
20. *bv* = 1;
21. **else if** round(*gbest_m*) = 1
22. *bv* = 0;
23. **end**
24. **end if**
25. *gbest_m* = bv; /* change the m^{th} variable in *gbest_m* for function evaluation */

Fig. 6.5 Pseudo codes framework of Improved KELS

6.4.1 Optimal Reactive Power Dispatch

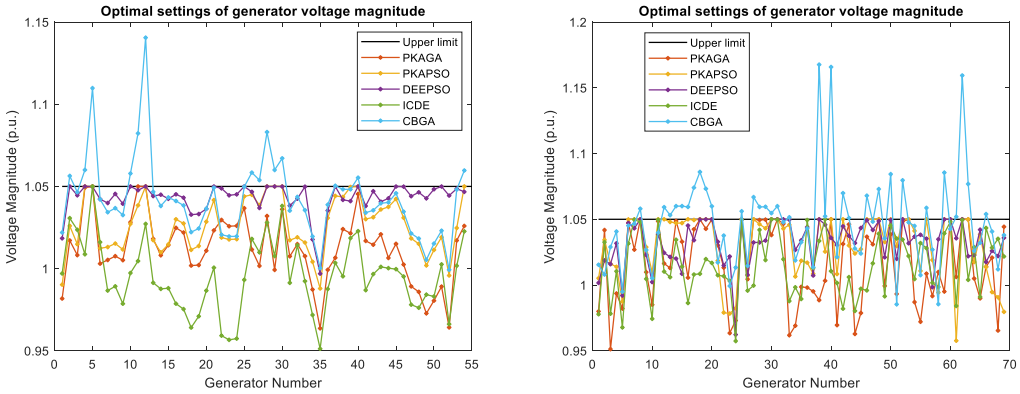
The objective of this optimal reactive power dispatch problem is to minimize the active power losses. The control variables (i.e. reactive power controls or VQ controls) comprise both continuous variables associated to the generator voltage magnitude, discrete variables associated to the OLTC, and binary variables associated to shunt compensations, as listed in Table 6.3. The formulation of ORPD is similar to the case 2 studied in chapter 5, but the integer and binary variables are considered, and the active power generation controls (i.e. P controls) are excluded. The exclusion of active power controls makes the optimal solution consistent with the economic principle, because of a unique feasible active power dispatch that is able to minimize both generation cost and active power losses at the same time may not exist, as studied in the previous work [253].

For illustrating the performance of proposed algorithm when solving the ORPD problems, the statistical result of 31 trials in terms of mean, best, worst and standard deviation (std) values and average computational time are given in Table 6.5. Note that the DEEPSO, ICDE and CBGA are the ranked algorithms of the benchmark test competition and their results are available in [254]. Moreover, in order to make a fair comparison, the ranked algorithms were re-run to reproduce the computational time. It can be seen from Table 6.5 that the statistical results of different test systems provide by the CBGA are overall the best. However, these results were achieved while violating the control variable limits, as shown in Fig. 6.6 (i.e. blue dot curve). This implies that the CBGA yields unreliable OPF solutions as some of the generator voltage magnitudes used to minimize the power losses were beyond the operational limits. Therefore, the results of CBGA are not discussed and compared further (i.e. the result discussions later do not include the CBGA).

Table 6.5 Statistical results of 31 trials obtained from the tested algorithms (ORPD)

System	Statistic	PKAGA	PKAPSO	DEEPSO	ICDE	CBGA
57	Mean	34.81	**24.87**	25.57	25.05	25.05
	Worst	307.34*	**25.23**	26.63	26.91	25.78
	Best	24.77	**24.64**	25.01	24.64	24.28
	Std.	50.62	**0.1798**	0.4114	0.4358	0.3869
	Time (s)	168.27	171.89	166.12	160.82	**151.71**
118	Mean	121.34	**118.13**	119.28	128.36	115.38
	Worst	127.55	**119.31**	127.36	135.69*	120.38
	Best	118.27	**117.41**	117.87	123.84	112.95
	Std.	2.4056	**0.5028**	1.7146	2.4546	1.7176
	Time (s)	504.34	445.09	419.20	**408.64**	429.33
300	Mean	1,371.91	**404.07**	414.62	8,669.71	374.47
	Worst	6,428.24*	582.06*	**562.46***	69,818.83*	382.13
	Best	391.38	**382.81**	391.43	400.08	371.65
	Std.	1,712.15	49.97	**42.04**	15,255.70	2.556
	Time (s)	3,228.23	2,961.45	2,754.21	**2,688.86**	2,702.32

Note: the results labelled with * mean that the constraint violations g shown in (6.5) are greater than 10^{-4}.



(a). IEEE 118 bus test system (b). IEEE 300 bus test system

Fig. 6.6 Optimal settings provided by different algorithms (ORPD)

Comparing the results of IEEE 57 bus test system, the differences among PKAPSO, DEEPSO and ICDE are insignificant, but the proposed PKAPSO outperforms others in terms of mean, worst, best and std values. In this particular test case, the developed PKAGA was unable to constantly find the feasible optimal solutions always within the specified constraints, thus its obtained mean, worst and std solutions given in the results table are greater than other algorithms. For the IEEE 118 bus test system, the PKAGA performed comparably with DEEPSO but better than ICDE; on the other hand, there was no infeasible solution detected for PKAGA. Considering the solution quality, the PKAPSO offers a slightly better solution including mean and best values than DEEPSO, while the consistency of PKAPSO is quite good (i.e. the worst solution and std values are much smaller than others). For the large-scale IEEE 300 bus test system, all algorithms produced infeasible solutions for some trials. Nevertheless, the mean and best optimal solutions of the proposed PKAPSO are clearly smaller than others. Moreover, the best optimal solution produced by PKAGA are marginally smaller than DEEPSO and much better than ICDE. In reference to the solution quality of all test systems yielded by PKAPSO, it can therefore conclude that the proposed PKAPSO is very reliable for solving the mixed-integer ORPD problems. Comparing the PKAGA and PKAPSO reveals that overall the PKAPSO provides better solution quality than PKAGA. This is somehow similar to the conclusion found in the previous optimization function tests that the PKAPSO could be able to yield more accurate solution and higher success rate (here is finding the feasible optimal solutions).

Regarding to the algorithms' average computational time of 31 trials, the proposed algorithms elapsed more computation time than the ranked algorithms, especially in the IEEE 300 bus test system. This is not unexpected because the proposed algorithms require additional efforts to build Kriging model and select best candidate solution before each function evaluation. However, directly comparing the total computation times of reaching the maximum FEs is somewhat inconclusive in reflecting the efficiency, because the algorithms actually converged very quickly while required much fewer FEs, as shown in Fig. 6.7. It can be seen from the convergence plots that the proposed PKAPSO obviously outperforms other algorithms in terms of requiring much fewer FEs to converge

to the optimum, and the PKAGA has better convergence speed than DEEPSO and ICDE in large-scale IEEE 118 and 300 test systems. In order to detailly analyse the convergence performance, Table 6.6 gives the convergent FEs, fitness (i.e. optimal solution) and the computation time, which were

Table 6.6 Convergent FEs, fitness and computation times (ORPD)

System	Index	PKAGA	PKAPSO	DEEPSO	ICDE	CBGA
57	FEs	9,300	**2,800**	12,000	7,000	3,100
	Fitness	358.97	**24.94**	477.80	27.85	27.94
	Time (s)	32.29	**9.63**	39.87	22.52	9.41
118	FEs	22,100	**4,800**	46,700	28,500	15,000
	Fitness	132.99	127.35	**120.99**	226.87	132.57
	Time (s)	111.46	**21.36**	195.77	116.46	64.39
300	FEs	50,400	**14,800**	78,000	inf	31,000
	Fitness	1,538.59	**487.54**	695.05	inf	1,169.58
	Time (s)	542.34	**146.09**	716.09	inf	279.24

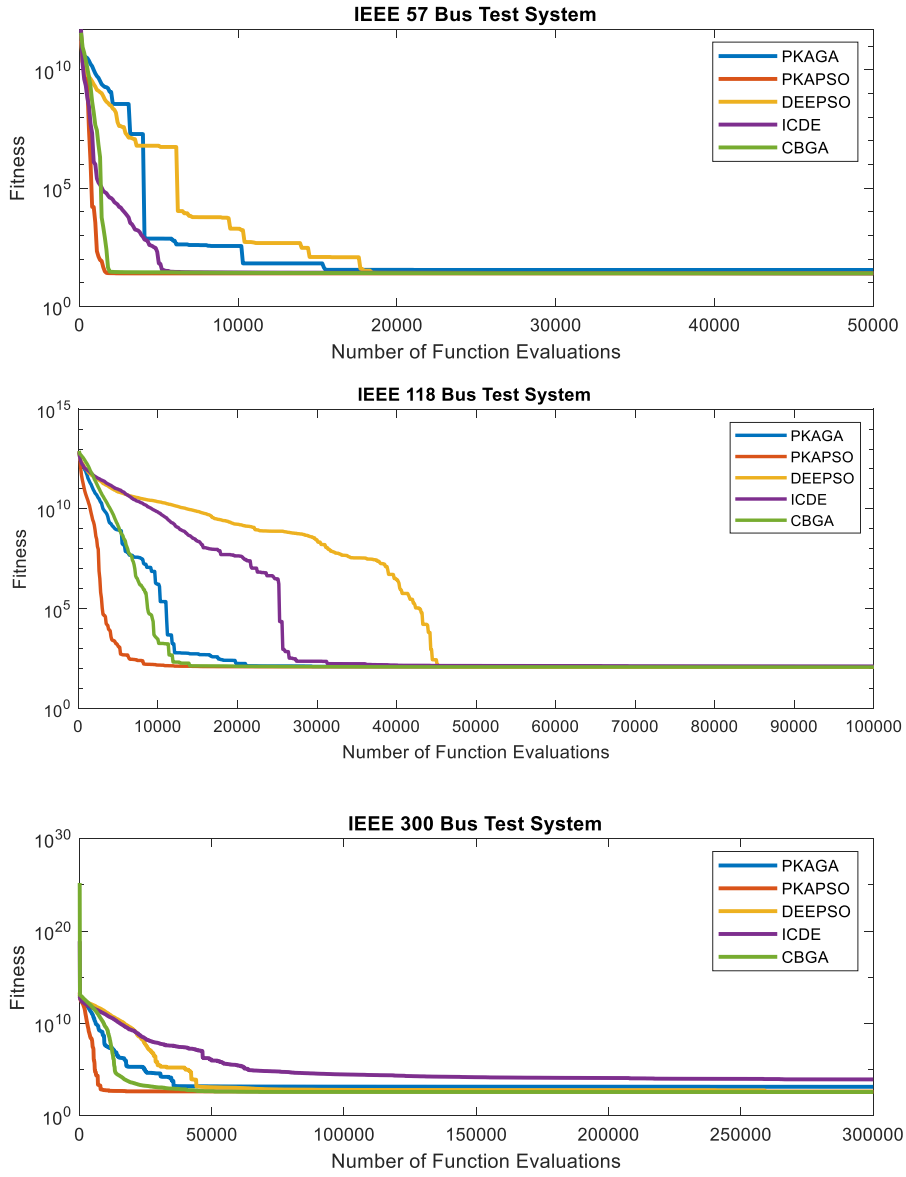


Fig. 6.7 Average convergence rate of 31 trials provided by different algorithms (ORPD)

recorded when the fitness was detected as not being updated (i.e. the improvement on average fitness was detected less than 1) for specified number of FEs (i.e. 10 for IEEE 57 and 118 bus test systems and 100 for IEEE 300 bus test system). "inf" means that the algorithm failed to converge, and the average fitness was continuously and largely improved (i.e. larger than 1) until reaching the maximum allowance of FEs. Note that the averagely convergent computation times presented in Table 6.6 were approximately calculated according to the total computation times of reaching maximum FEs given in Table 6.5. That is, the average computation time per FE times the average number of convergent FEs. A more detailed inspection of Table 6.6 reveals that PKAPSO has shown its superior convergence performance regarding to fewer FEs, better fitness (except slightly larger fitness of IEEE 118 bus test system than DEEPSO) and shorter convergence times than others. The PKAGA achieved better convergence speed with shorter elapsed computational time and smaller number of FEs than DEEPSO, although it produced larger convergence solutions in IEEE 118 and 300 bus test systems. The DEEPSO and ICDE have shown to be reliable in finding optimal solution, but it required a large number of function evaluations and is somehow computationally inefficient. From the convergence point view, the proposed PKAPSO is very robust and computationally efficient in finding feasible global optimum for ORPD problems. Note that the convergence performance of an algorithm is not insignificant, for example, a stochastic based algorithm could run many trials rather than single trial to find out the best solution within the limited time if it has good convergence performance. Because the stochastic algorithm usually produces inconsistent solutions in different trials, especially when dealing with large-scale optimization problems, it may be too risky and greedy to find the global optimum relying on single trial.

In order to further verify the effectiveness of using pre-selection Kriging to improve the evolutionary based algorithm when solving OPF problem, the proposed pre-selection strategy was applied to the DEEPSO and ICDE. It was found that the ICDE provided better average fitness and faster convergence speed when the Kriging was used. Besides, the Kriging improved the convergence rates of DEEPSO, but it resulted in larger average fitness values for both IEEE 118 and 300 bus test systems. This may not imply that the Kriging would degenerate the performance of DEEPSO, rather it means that the fitness value of the worst trial was assigned larger penalty value due to the infeasible solution, while the DEEPSO also provided infeasible solutions for some trials. For example, the worst fitness value (for 300 bus system) obtained from DEEPSO and Kriging assisted DEEPSO were 562.46 and 7622.04, respectively, both the solutions violated the inequality constraints and may be infeasible; whereas Kriging helped to obtain a slightly smaller optimum in the best trial which was 389.33. For a better view and comparing the performance between original DEEPSO and ICEDE and Kriging assisted versions, the convergence plots are given in Fig. 6.8. The convergence plots indicate that the proposed Kriging pre-selection strategy may be a potential and effective approach for improving those well-developed algorithms, particularly the computationally efficiency, although the DEEPSO was not constantly improved in all test cases of this study.

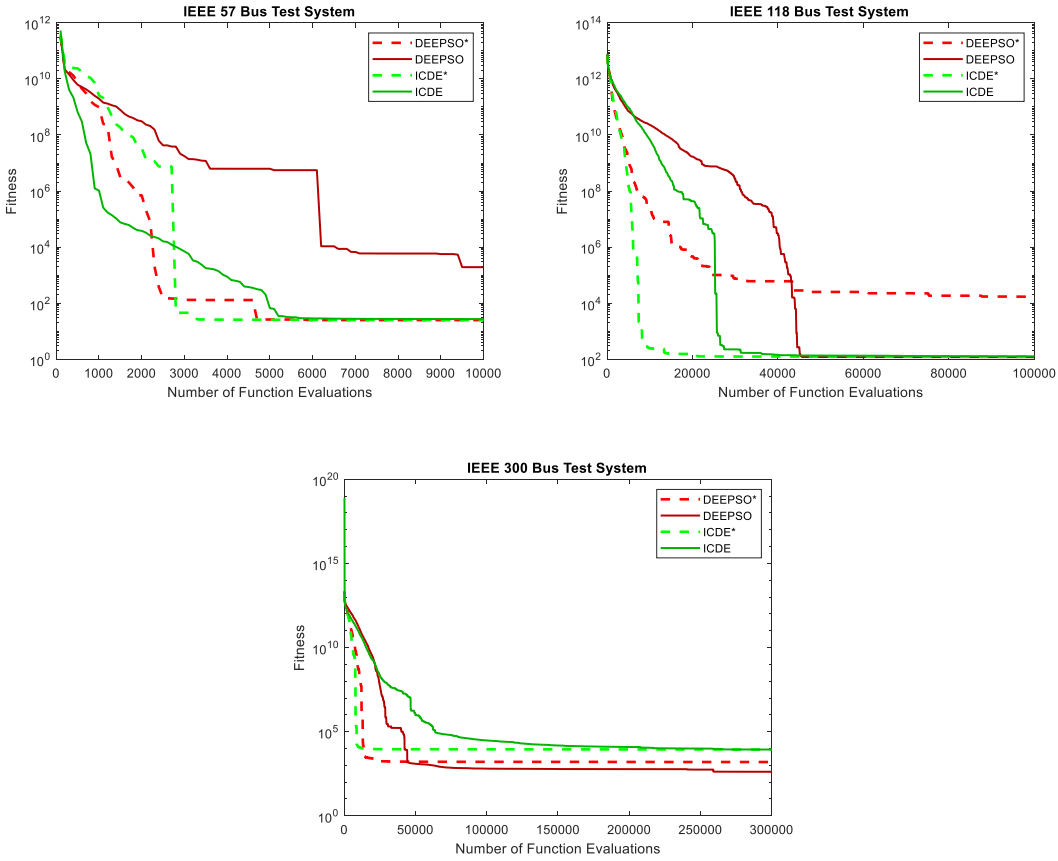
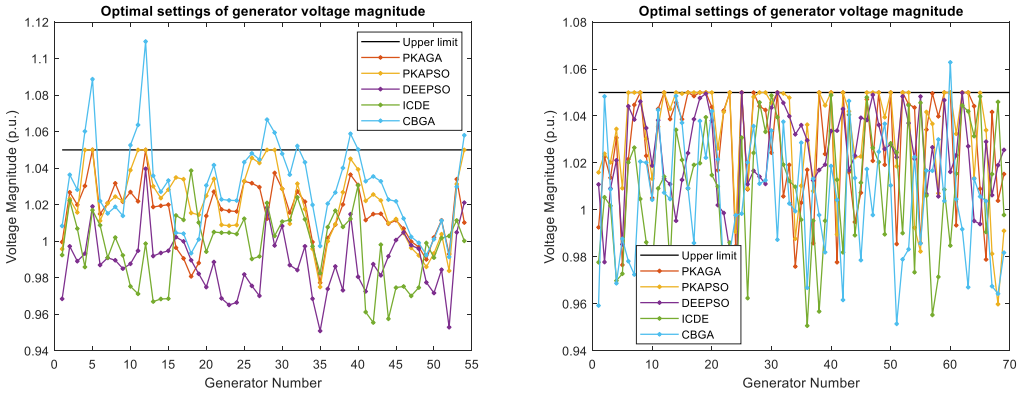


Fig. 6.8 Average convergence rates of 31 trials performed by ranked algorithms and Kriging assisted algorithms ("*" means the algorithm is assisted by proposed Kriging pre-selection)

6.4.2 **Optimal Active and Reactive Power Dispatch**

This optimal active and reactive power dispatch problem aims at minimizing the dispatch cost using a quadratic cost function, same as in the case 1 studied in chapter 5. Namely, it optimizes the active and reactive power outputs to achieve minimum generation cost. The control variables of this problem not only include the continuous, discrete and binary variables used in ORPD problem, but also considers the other continuous variables which most directly influence the generation costs – active power output of generators. In practical power system operation, the system components (i.e. variables) may be not all available for system operator to control, particularly when the system operation is close to the operating time and the components, for example shunt compensation, are locally controlled. Without practical guidance, it may be difficulty to decide appropriate control variables, therefore in study model it usually use an "optimistic" OPF formulation that assumes all variables are controllable, while this test case formulated by the test bed is no exception. Moreover, the reactive power controls may not be considered in practice due to their insignificant impact on operational efficiency, but the academic research takes these into account because they are capable of reducing active power losses and indirectly influencing the total generation costs.



(a). IEEE 118 bus test system (b). IEEE 300 bus test system

Fig. 6.9 Optimal settings provided by different algorithms (OARPD)

For the fairly comparisons, the results of CBGA illustrated here are also not being discussed because the generator voltage magnitude constraints were not considered by the CBGA, as shown in Fig. 6.9 (i.e. blue dot curve). The mean, worst, best and standard deviation values and average computation time of 31 trials obtained from the proposed algorithms and the ranked algorithms of this benchmark competition are given in Table 6.7. For the small-scale IEEE 57 bus test system, it can be seen that the DEEPSO provides better mean and best solutions (not significant) than the proposed algorithms; however, both the developed PKAGA and PKAPSO offers superior statistical solutions than ICDE especially the PKAPSO algorithm. For the large-scale IEEE 118 and 300 bus systems, the PKAGA and PKAPSO considerably outperformed the ranked algorithms regarding to

Table 6.7 Statistical results of 31 trials obtained from the tested algorithms (OARPD)

(OARPD)

System	Statistic	PKAGA	PKAPSO	DEEPSO	ICDE	CBGA
57	Mean	41,714.95	41,706.43	**41,697.52**	41,739.97	41710.51
	Worst	41,753.68	**41,721.96**	41,727.82	41,805.35	41729.98
	Best	41,695.26	41,690.02	**41,688.66**	41,702.84	41693.32
	Std.	11.1581	8.2261	**7.9393**	27.7701	11.058
	Time (s)	160.45	156.97	156.75	157.11	**146.15**
118	Mean	135,116.45	**135,054.67**	135,892.65	154,244.46	135108.57
	Worst	135,244.06	**135,117.93**	136,381.65	173,785.34	137704.12
	Best	135,050.49	**135,012.63**	135,492.35	136,238.11	134968.83
	Std.	48.8694	**24.3459**	224.09	12,325.52	483.54
	Time (s)	560.42	500.15	440.08	434.09	**429.98**
300	Mean	721,924.56	**721,143.11**	723,042.22	740,112.65	1.828E+13
	Worst	729,133.13	**723,996.68**	736,527.31	754,493.47	5.305E+13
	Best	720,496.11	**720,428.33**	721,377.59	722,608.84	7.2036E+5
	Std.	1,912.43	**1,090.82**	2,669.59	6,205.42	1.588E+13
	Time (s)	3,763.24	3,262.48	3,131.29	**2,864.71**	4,523.31

the mean, worst, best and std solutions, although the average elapsed computational time was longer. Besides, again the results of PKAPSO were better than the PKAGA. Considering the excellent solution quality provided by PKAGA and PKAPSO, it could then conclude that the proposed algorithms are very reliable for finding feasible optimal settings for OARPD problem while providing minimum generation cost. Overall, regarding to the performances of using proposed algorithms to solve different OPF problems, the generation costs obtained from the proposed algorithms comparing with other ranked algorithms are distinctly lower, while the minimum losses provided by the proposed algorithms are comparable with others. It can therefore suggest that the proposed algorithms are more capable and favourable for solving both ORPD and OARPD problems.

Fig. 6.10 demonstrates that the proposed PKAGA and PKAPSO converge to the optimum using much fewer necessary number of FEs than others. In order to detailly analyses the convergent

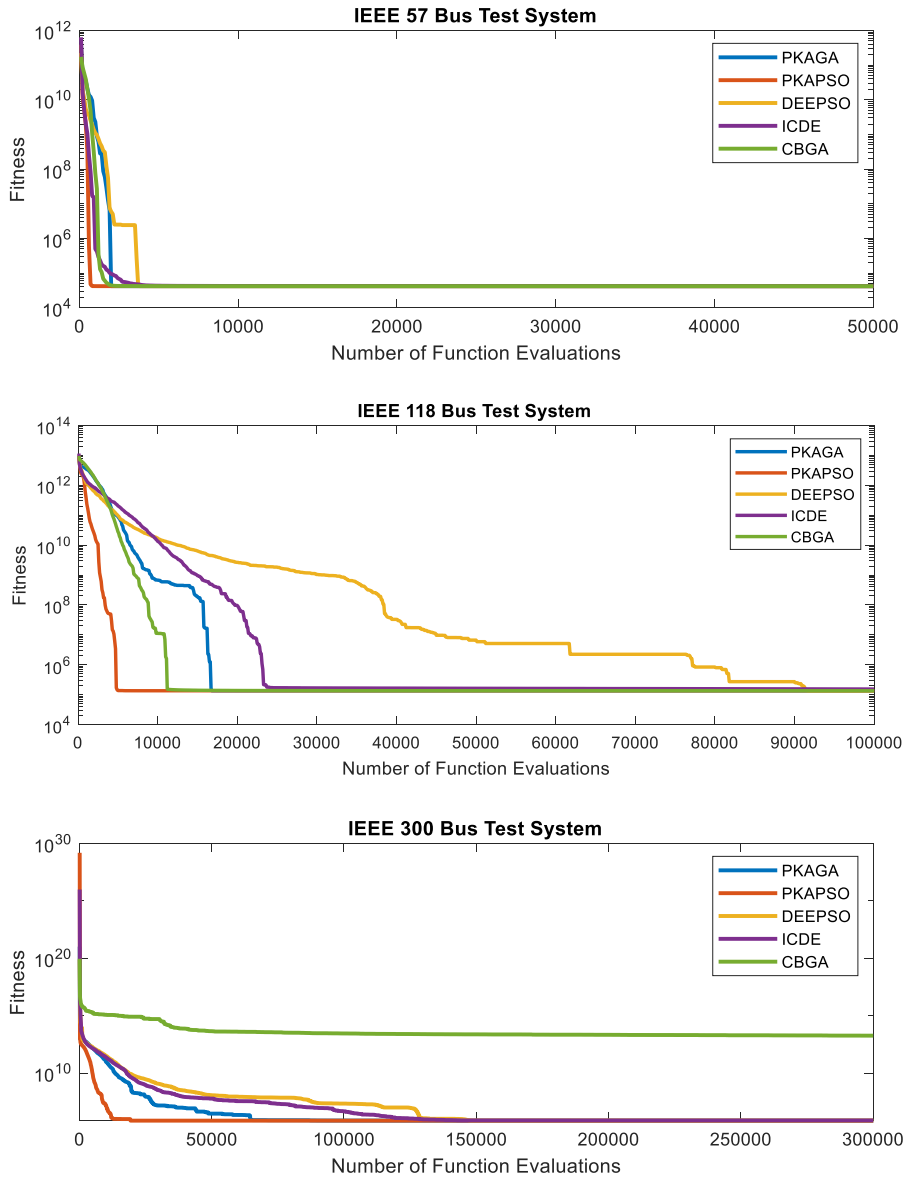


Fig. 6.10 Average convergence rate of 31 trials provided by different algorithms (OARPD)

Table 6.8 Convergent FEs, fitness and computational times (OARPD)

System	Index	PKAGA	PKAPSO	DEEPSO	ICDE	CBGA
57	FEs	7,400	**5,300**	12,900	24,300	20,300
	Fitness	41,717.41	**41,707.97**	41,707.67	41,787.81	41,725.81
	Time (s)	23.75	**16.64**	40.44	76.36	59.34
118	FEs	32,300	**12,400**	inf	inf	858
	Fitness	135,166.11	**135,126.45**	inf	inf	135,264.38
	Time (s)	181.01	**62.02**	inf	inf	3.69
300	FEs	34,800	**25,000**	inf	243,400	11,400
	Fitness	1.293E+7	**721,223.58**	inf	740,791.89	1.160E+15
	Time (s)	311.09	**271.87**	inf	2,324.21	171.88

performance of proposed algorithms, the averagely convergent fitness, necessary FEs and CPU times are given in the Table 6.8. Note, these convergent results are also obtained using the same approach as in ORPD problem (i.e. the fitness was detected as not being updated for specified number of FEs – 10 for IEEE 57 and 118 bus test systems and 100 for IEEE 300 bus test system). The DEEPSO and ICDE failed to converge to the optimum for some test systems (according to the criterion as used in last ORPD problem), whereas the developed algorithms especially PKAPSO outperformed others in terms of much shorter convergence time, much fewer required FEs and much better fitness. These results may imply that the developed algorithms are also very robust and computationally efficient

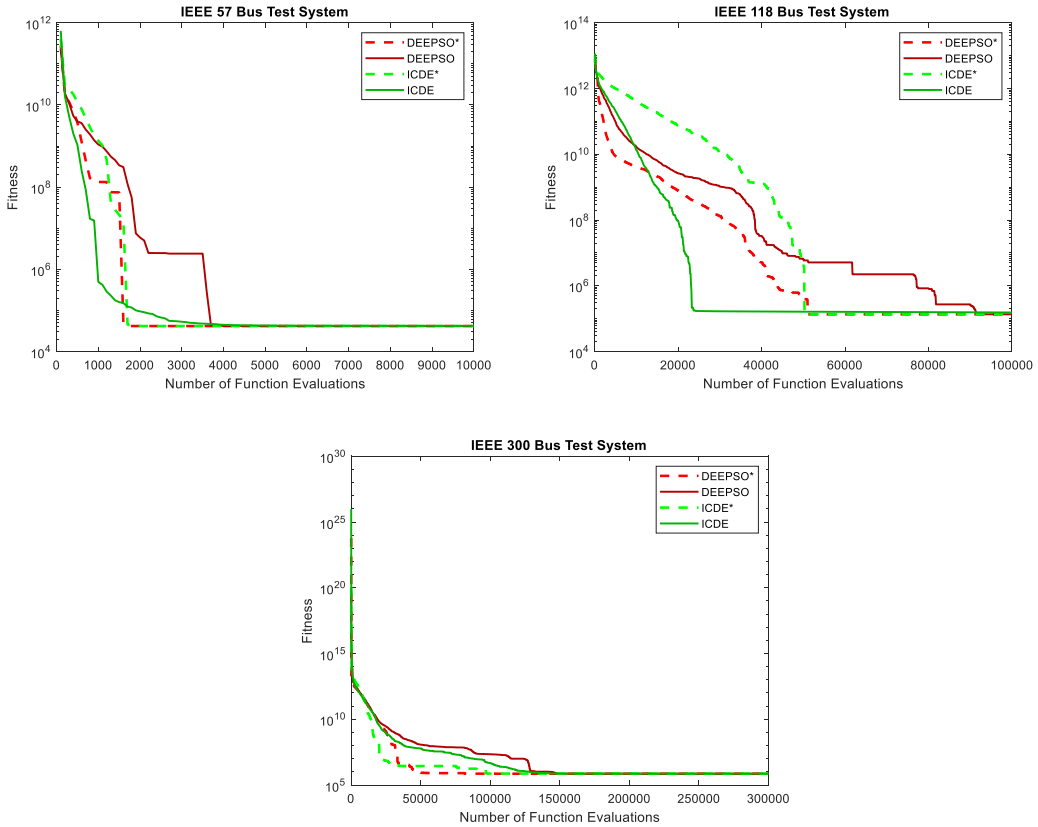


Fig. 6.11 Average convergence of 31 trials performed by ranked algorithms and Kriging assisted algorithms ("*" means the algorithm is assisted by proposed Kriging pre-selection)

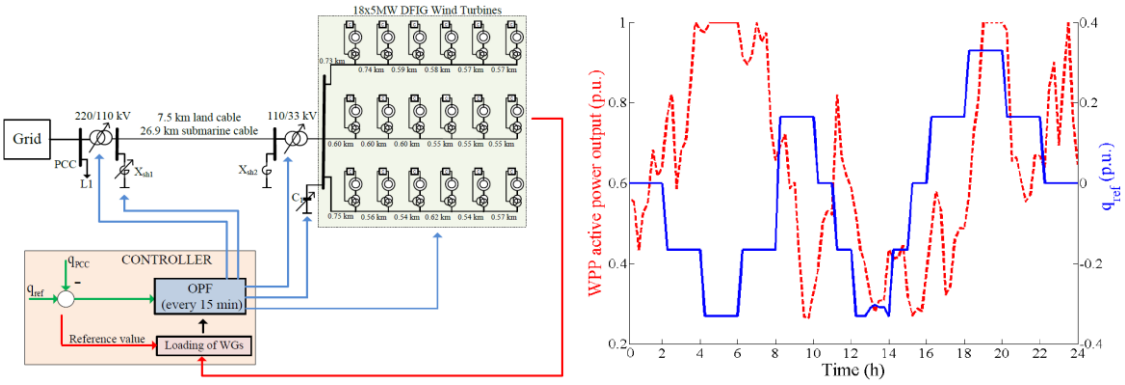
for solving the OARPD problem containing more optimization variables. Moreover, it was noticed that the DEEPSO and ICDE performed much better convergence rate and convergent optimum in most test systems (except ICDE on IEEE 118 bus test systems) when the proposed Kriging pre-selection strategy was associated to the algorithms, as shown in Fig. 6.11. This reveals that the proposed approach of using Kriging is also reliable for improving other advanced evolutionary algorithm further when solving economic dispatch related OPF problems.

6.4.3 Offshore Wind Power Plant

In addition to the traditional OPF problems studied previously, the test bed also provides an OWPP system for testing the algorithm's performance when solving online optimal reactive power control problem (i.e. ORPD). The layout of the OWPP system is illustrated as in Fig. 6.12 (a) [252], which consists of 18 wind power generators, 2 shunt reactors – one from offshore side is not allowed to adjust and another one connected to the grid side is adjustable, two stepwise OLTCs and one stepwise regulated shunt capacitor. This test focuses on minimizing the active power losses through adjusting wind generator reactive power outputs (continuous), OLTC (discrete), shunt capacitor (discrete) and shunt reactor (continuous). Moreover, this reactive power control problem assumes that the active power outputs profile corresponding to all 15 minutes interval points throughout 24 hours are accurately forecasted, while the controller (i.e. OPF optimizer) needs to continuously fulfil the stepwise changed reactive power demands (i.e. 96 scenario) at Point of the Common Coupling (PCC), as shown in Fig. 6.12 (b) [252].

It has been stated by the guidebook of test bed that some of the task in the 96 scenarios are hard-to-solve optimization problem. Indeed, it was found that both proposed algorithms and ranked algorithms were unable to constantly find the feasible solutions under some certain scenarios for 31 trials. It is difficult to compare the statistical performance of 31 trials one scenario by one scenario, because the algorithm performed better for some certain scenarios and worse for others, as shown in Fig. 6.13. Nevertheless, it was found that the differences among the minimum losses of scenario 17-24 obtained from the test algorithms are distinct while other scenarios are comparable with tiny differences. Under this consideration, here the total active power losses of 96 scenarios obtained from the best trials are given in Table 6.9. Even through the best trial was considered, the PKAGA, ICDE and CBGA were unable to find the feasible optimum under some hard-to-solve scenarios for at least one trial. Comparing the total active power losses, the proposed PKAPSO perform as good as ranked DEEPSO (marginally better), while no infeasible solution is found for the best trials. When Kriging pre-selection was employed to DEEPSO, it produced slightly higher total losses, however, the infeasible solutions issues of ICDE was resolved by using Kriging. This is somehow consistent with the Kriging performed in the ORPD problem that the Kriging is helpful for ICDE while offering slightly worse solution when applying to DEEPSO. Finally, according to the solution quality, the

proposed PKAPSO again is much better than the PKAGA, while the Kriging pre-selection is also a capable and potential approach for improving the well-developed algorithms.



(a) (b)

Fig. 6.12 (a). Offshore wind power plant with reactive power management system; (b). Active and reference reactive power outputs of wind power plant

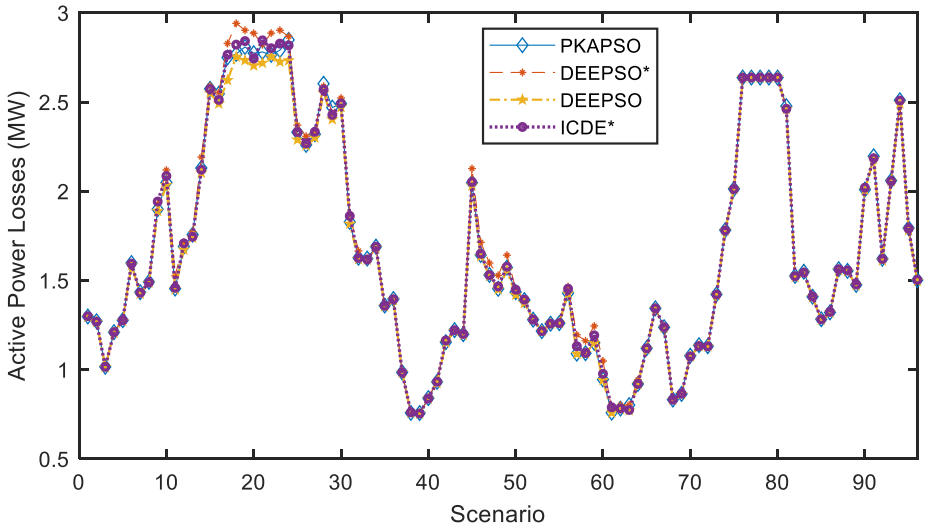


Fig. 6.13 Active power losses of the best trial at each scenario (OWPP)

Table 6.9 The total active power losses of 96 scenarios from the best trial

Algorithms	Losses (MW)	Infeasible scenarios
PKAGA	1.1660E+10	49 – 56, 74 – 80
PKAPSO	160.88	N/A
DEEPSO*	162.80	N/A
DEEPSO	**159.58**	N/A
ICDE*	161.24	N/A
ICDE	8.4061E+5	52 – 55, 76 – 80
CBGA*	4.3840E+9	75 – 81
CBGA	6.5263E+7	73 – 81

6.5 Summary

In this chapter, a pre-selection Kriging strategy, mitigating the "Curse of Uncertain" and while maximizing the "Blessing of Uncertain", has been proposed to improve the convergence performance of evolutionary based algorithm for efficiently finding better OPF solutions. To enhance the exploitation and exploration capability of evolutionary algorithm, two improved evolutionary operators – KELS and KOEDLS – supported by an efficient Kriging pre-selection have been introduced. In order to balance the exploitation and exploration when selecting different proposed evolutionary operators, as well as selecting promising solutions based on WMEI criterion, an award-based method has been introduced. An efficient award approach has been proposed to adaptively control the use of the proposed operators, as well as assign the weight to expected improvement to balance the exploitation and exploration when selecting the best candidate during the optimization process. The "Iteratively Rolling Replacement" approach has been proposed to build a reasonable pseudo global Kriging model with a constant sample size and following evolutionary trajectory, while alleviating the computational cost issue associated to the Kriging estimation. Finally, two algorithms combining the proposed strategy with modified canonical GA and PSO, named PKAGA and PKAPSO, respectively, have been developed.

In order to verify the effectiveness of proposed approaches, the individual-based and the pre-selection strategy were applied to some common GA and PSO variants and tested and compared using the benchmark optimization functions. The results have demonstrated that GA and PSO and their variants with the assistance of pre-selection strategy have achieved better convergence performance. In addition, the OPF test bed, including OARPD and ORPD problems of IEEE 57, 118 and 300 bus test systems and OWPP system, developed by IEEE working group was used to verify the applicability and effectiveness of the developed PKAGA and PKAPSO. By comparing the OPF results from PKAGA and PKAPSO with the ranked algorithms from benchmark competition, it has been found that PKAPSO offers overall best performance by consistently finding lower generation costs or smaller active power losses using shorter computing times. Although the other proposed algorithm PKAGA is not as good as PKAPSO, it is still better than the cited algorithms in some cases. Moreover, the convergence performance (particularly in OARPD problems) of the ranked algorithms from benchmark competition has been improved by using the Kriging pre-selection strategy. Finally, according to the solution quality yielded by PKAPSO and PKAGA and the improvement of ranked algorithms made by using Kriging pre-selection, it is therefore concluded that the developed pre-selection strategy is a promising tool for improving the performance of stochastic based algorithms when solving complex OPF problems.

Chapter 7 Conclusion and Recommendations for Further Work

7.1 Conclusions

Optimal power flow is an important tool in the field of power system operation and energy management, but the traditional derivative-based optimization method is very difficult or impossible to provide the global optimal solution for full AC power system model, which may unnecessarily cost billions of dollars per year. Although various evolutionary based algorithms have been proposed in the academic research to address this issue, their applications in practical power system operation have been limited due to the impractically long execution times. This mainly motivated this work to develop a technique, which could maintain the optimality while yielding shorter computational time, to advance the application of evolutionary based algorithm in practice and save large amount of operational costs. According to the previous works of Southampton Group, it was found that the surrogate modelling method could be a potential technique to improve the computational efficiency of evolutionary based algorithms, and it was not being employed in the context of power system optimization before. However, surrogate assisted evolutionary algorithms were originally developed for those small-scale and computationally expensive optimization problems, whereas power system optimization is computationally cheaper and tends to a have large number of variables, thus the well-developed approaches in the literature cannot be applied to solve OPF problems directly. Therefore, the focus of this work was to develop and investigate the reliable and robust surrogate based technique to improve the computational performance of evolutionary based algorithm when solving OPF problems.

The objectives of this work specified in chapter 1 were all accomplished. In order to clearly understand OPF, the practical concerns of OPF used in industry, the fundamental details of OPF formulation, and the current OPF solution algorithms have been critically reviewed to introduce and discussed their related technique issues. It was found that the OPF problem based on full AC power flow model is naturally a nonlinear and nonconvex optimization problem with both continuous and discrete variables. Nevertheless, throughout the literature, the current OPF solution techniques either scarify the solution quality or computational efficiency when dealing with this kind of difficulty problem. However, the use of new technologies dramatically increases the complexity of OPF problem, and the power system industry urges the need of a more accurate and faster OPF algorithm because it can greatly improve the system reliability and save the operational costs. To figure out which surrogate modelling technique is most suitable for this work, three well-established methods – including PR, RBF and Kriging – have been introduced. Based on their numerical modelling performance, the Kriging was selected as the main surrogate method for this work because it is able

to build the accurate model using a small number of samples and capable of balancing exploration and exploration. For the purpose of using Kriging to select the most promising infill point to update the model as well as finding the global optimum, a WMEI criterion has been introduced and studied. It has been demonstrated that the weight of EI has significant impact on the exploitation and exploration when Kriging selecting the infill point. The Euclidean distance-based and award-based weight strategy have been developed to adaptively calculate the weight value while achieving a balance between exploitation and exploration. To combine the Kriging with evolutionary algorithm while improve the computational performance, the EGO technique was introduced and studied and the SAEC technique have been reviewed. It was demonstrated that the EGO technique using additional optimization solver to find the WMEI infill point as well as the global optimum is computationally inefficient due to the computational cost of Kriging estimation associated with the large matrix inversion and the interactively increasing sample set. To alleviate this computation issue and improve the computational efficiency, the "Better-Adoption and Worse-Abandonment" strategy and the "Iteratively Rolling Replacement" sample update strategy with constant sample size have been proposed to build local and pseudo-global Kriging model, respectively. Moreover, it was found that the SAEC technique is more suitable for this work. However, the evolution control of SAEC may suffer from the "Curse of Uncertainty" that the algorithm may be easily misled by the estimated false optimum, whereas the pre-selection based SAEC may not be computationally efficient as the Kriging estimation is needed before each function evaluation.

Since the SAEC techniques are not applied to OPF before, it is difficult to make the right judgement on which strategy is the best for managing the Kriging in evolutionary computation when solving OPF problems. Therefore, these two strategies have been investigated and developed in this work. First, an improved individual-based evolution control strategy has been proposed to avoid the "Curse of Uncertainty" by not including the estimated fitness in the evolutionary optimization process. The individual-based Kriging assisted algorithms, namely KAGA and KAPSO, have been developed to improve the original GA and PSO, while a Gaussian mutation based ELS operator was adopted to improve the algorithms' exploration capability. This kind of Kriging strategy, however, may overlook the good solutions in the population and does not fully utilize the parallel computing nature of evolutionary algorithm, since only one statistically promising solution is selected to be evaluated using real objective function at each iteration. Under these concerns, another pre-selection Kriging strategy that is able to maximize the "Blessing of Uncertainty" and utilize the parallel computation property has been proposed, while two improved pre-selection Kriging assisted algorithms, namely PKAGA and PKAPSO, have been developed. Furthermore, in these two algorithms, two additional KELS and KOEDLS operators supported by the Kriging estimation have been suggested to improve the algorithms' exploitation and exploration capability as well as the convergence performance. Finally, the developed KAGA, KAPSO, PKAGA and PKAPSO were tested on the numerical optimization functions and OPF problems to verify their performance in terms of solution quality and computational efficiency. Comparing the simulation results between

the proposed algorithms and the reported algorithms from the literature suggests that the proposed Kriging strategies are very promising for improving the performance of evolutionary algorithms when solving OPF problems. Particularly, the pre-selection strategy was more reliable than the individual-based strategy, Kriging strategies applying to PSO offered better results than that applying to GA, and the PKAPSO overall outperformed other algorithms.

The main contributions of this work can be identified as follows:

- First time introduction of Surrogate Assisted Evolutionary Computation technique for solving optimal power flow problems. The proposed individual-based technique uses a surrogate model to reduce the total function evaluations therefore achieving important improvements on computation efficiency (i.e. CPU time reduction). Two flavours of this technique have been implemented here first the Kriging Assisted Genetic Algorithm and then Kriging Assisted Particle Swarm Optimization. Both implementations have been developed and demonstrated improvements in the solution accuracy and computational efficiency as compare with the evolutionary based algorithms used normally for solving optimal power flow problems. Moreover, the proposed strategy could be potentially combined with other evolutionary based algorithms to improve their performance.
- Two method (i.e. “Better-Adoption and Worse-Abandonment” strategy and “Iteratively Rolling Replacement” strategy) to keep the constant sample size have been proposed to control the size of correlation matrix therefore alleviating and maintaining the computational cost of constructing Kriging model within certain limits. While the new methods propose to use the selected infill point to replace the trivial point from the sample set. It should be noted that, large matrix inversion is the main bottle neck of using Kriging to solve large scale optimization problem and it perhaps the reason why it hasn’t been applied to OPF problem so far.
- The concept of local and pseudo-global surrogate model has been introduced and used to alleviate the computational cost and fidelity issues associated to the Kriging. These suggest that the Kriging model should be build based on the trajectory of evolutionary optimization process.
- Two novel adaptive weight strategies have been proposed to balance the exploitation and exploration when using Kriging to select the infill point. They are able to dynamically provide the EI with different weight to adjust the emphasis on local and global search, while improving the solution quality and computational efficiency of the algorithm. The Euclidean distance-based strategy depends on space distance between each unknown solution to current best solution, whereas the award-based strategy relies on whether the current solution is updated.
- A pre-selection strategy has been proposed to improve the convergence performance of evolutionary based. Unlike the idea in individual-based strategy that is to directly reduce the

number of function evaluations at each iteration, pre-selection strategy speeds-up the algorithm via improving the convergence performance. Two algorithms, named PKAGA and PKAPSO, have been developed and been demonstrated with significant improvement on the convergence performance. Moreover, the GA and PSO and the algorithms from OPF benchmark competition were successfully improved when the pre-selection Kriging was used. Considering the solution quality, this strategy is very promising for improving other evolutionary based algorithms when solving OPF problems.

- The mutation strategy and factor analysis operators supported by Kriging have been proposed to enhance the exploitation and exploration capabilities of evolutionary algorithm. The necessary function evaluations needed by original factor analysis are significantly reduced by using Kriging.
- The award method associated to the developed algorithms has been proposed to control the use of the proposed operators and to assign the weight to expected improvement.

7.2 Recommendations for Further Work

The general academic goal of this work was to develop a new technique to advance the application of evolutionary based algorithms to solve practice OPF problem accurately and efficiently. It has been demonstrated that the proposed Kriging strategies and the associated novel approaches are capable of improving the computational efficiency of evolutionary based algorithms as well as the solution quality. However, there are still remaining challenges for further research and improvement. The following recommendations may be put forward for further work:

- The accuracy of any surrogate methods is always a critical concern, while it is very difficult to build a high-fidelity model when dealing with nonconvex and large-scale problem due to the "Curse of Dimensionality". According to the study in appendix B3, the hyperparameter $\boldsymbol{\theta}$ is one of the key elements that influences the accuracy of Kriging. However, the process of finding optimal $\boldsymbol{\theta}$ is not nontrivial and usually increases the computational cost of Kriging estimation. Nevertheless, this work set different empirical values to $\boldsymbol{\theta}$ in different test cases to avoid additional computation effort while achieving reasonable Kriging model. Therefore, it is not insignificant to develop a computationally efficient method to find the optimal $\boldsymbol{\theta}$ and improve the accuracy level of Kriging model.
- In addition to the hyperparameter, the sample is also very important for achieving high accuracy Kriging estimation. If the samples used to build the Kriging model are all far away from the estimation point, it is impossible to provide an accurate estimation. Although the proposed local and pseudo-global Kriging model are able to track the trajectory of evolutionary process to build the acceptable Kriging model while the estimation points are highly correlated

with the samples, many evaluated samples are abandoned, and most remaining samples may be gathering around a local optimum (especially in "Better-Adoption and Worse-Abandonment" strategy). Hence, it may need to store all the evaluated samples in the data file while using the potential clustering technique (e.g. the well-known K-mean clustering) and Cross-validation method to select the most relevant samples for each estimation. However, it has to be emphasized that the proposed sample strategies are computational efficient, and it should be careful with the computation cost when using other techniques to select the samples. The principal component analysis may be a potential technique to reduce the size of correlation matrix to reduce the computational cost.

- Due to the sample size is associated to the computational cost of Kriging estimation and correlation matrix, this work used the constant sample size. However, the sample size defined for different test cases was roughly set equally to the population/swarm size. It is necessary to develop an adaptive approach for defining the sample size as well as saving computational cost and maintaining modelling quality. To further improve the computation efficiency of Kriging estimation when dealing with the large-scale problem, the dimension reduction techniques, such as Principle Components Analysis and Partial Least Squares [255], could be used to reduce the size of correlation matrix.
- The proposed individual-based strategy is somehow greedy as only one individual or particle is selected for function evaluation, some good solutions, however, may be overlooked during the optimization process. Therefore, it could be further improved by developing an adaptive approach to intelligently select the use of individual, population and generation control. Note that the "Curse of Uncertainty" should be properly addressed.
- On the other hand, the proposed pre-selection strategy is considered to be conservative rather greedy, because it uses the Kriging to screen out promising solution before each function evaluation. According to the test results presented in appendix B7, the candidate solution size clearly affects the performance of pre-selection Kriging assisted algorithms, despite sometimes insignificant. However, the smaller the size of candidate solutions the lower the computation cost. Intuitively, sometimes the pre-selection may not be necessary and thus a technique (e.g. Reinforcement Learning) that is able to decide when and where to use the Kriging pre-selection is needed. Moreover, it is possible to combine the proposed individual-based and pre-selection strategies together to achieve better performance in terms of better solution quality and higher computation efficiency.
- In the proposed award-based weight strategy, it assumes that the weight value of EI is strongly associated with the exploitative and explorative operators in the evolutionary algorithm, enabling the balance between exploitation and exploration. However, it was not thoroughly studied in this work, and it may need to be further investigated to analyse its theoretical background. Besides, both the award used to assign weight and the assigned weigh value could be further improved.

- Furthermore, the proposed algorithms need to be further verified on more complex OPF problems, such as Transient-Constrained OPF, Security-Constrained OPF, Smart Grid OPF, to prove their effectiveness and performance. As discussed in chapter 2, the convex relaxation techniques are able to convexify the OPF problem, but they cannot easily deal with the infeasible solution issue. Nonetheless, the proposed algorithms could be used to solve the OPF problems in the form of relaxed convex, since they could easily address the OPF constraints while Kriging can use small number of samples to accurately model the convex problem. Finally, it is worthy to further analyse the performance of the proposed Kriging strategies when applying to different evolutionary algorithms, using different infill point criterions, and replacing by different surrogate modelling techniques.

List of Publications

Conference

Z. Deng, M. D. Rotaru and J. K. Sykulski, "A study of evolutionary based optimal power flow techniques," 2016 51st International Universities Power Engineering Conference (UPEC), Coimbra, pp. 1-6.

Z. Deng, M. D. Rotaru and J. K. Sykulski, "Harmonic Analysis of LV distribution networks with high PV penetration," 2017 International Conference on Modern Power Systems (MPS), Cluj-Napoca, pp. 1-6.

Journal

Z. Deng, M. D. Rotaru and J. K. Sykulski, "Kriging Assisted Surrogate Evolutionary Computation to Solve Optimal Power Flow Problems," in IEEE Transactions on Power Systems, vol. 35, no. 2, pp. 831-839, March 2020.

Z. Deng, S. Xiao, M. D. Rotaru and J. K. Sykulski, “Optimal Power Flow by Pre-selection Kriging Assisted Evolutionary Computation” (submitted to IEEE Transactions on Power Systems and under review).

List of References

[1] Ardelean M and Minnebo P, “A China-EU electricity transmission link Assessment of potential connecting countries and routes,” *JRC Sci. Policy Rep.*, pp. 1–99, 2017.

[2] G. Chen, M. Hao, Z. Xu, A. Vaughan, J. Cao, and H. Wang, “Review of high voltage direct current cables,” *CSEE J. Power Energy Syst.*, vol. 1, no. 2, pp. 9–21, 2015.

[3] J. W. Feltes, B. D. Gemmell, and D. Retzmann, “From smart grid to super grid: Solutions with HVDC and FACTS for grid access of renewable energy sources,” *IEEE Power Energy Soc. Gen. Meet.*, pp. 1–6, 2011.

[4] A. Alassi, S. Bañales, O. Ellabban, G. Adam, and C. MacIver, “HVDC Transmission: Technology Review, Market Trends and Future Outlook,” *Renew. Sustain. Energy Rev.*, vol. 112, pp. 530–554, 2019.

[5] Parliamentary Office of Science and Technology, “UK Electricity Networks,” *POSTnote*, no. 163, pp. 1–4, 2001.

[6] T. Basso, “IEEE 1547 and 2030 Standards for Distributed Energy Resources Interconnection and Interoperability with the Electricity Grid IEEE 1547 and 2030 Standards for Distributed Energy Resources Interconnection and Interoperability with the Electricity Grid,” *Nrel*, no. December, p. 22, 2014.

[7] International Energy Agency, “World Energy Outlook 2017,” Paris, 2017.

[8] M. Cain, R. O’Neill, and A. Castillo, “History of Optimal Power Flow and Formulations (OPF Paper 1),” *US Fed. Energy Regul. Comm. Staff Tech. Rep.*, pp. 1–36, 2012.

[9] T. E. Dy-Liacco, “Control centers are here to stay,” *IEEE Comput. Appl. Power*, vol. 15, no. 4, pp. 18–23, 2002.

[10] E. Vaahedi, “Practical Power System Operation,” *John Wiley Sons*, pp. 105–128, 2005.

[11] F. F. Wu, K. Moslehi, and A. Bose, “Power System Control Centers: Past, Present, and Future,” *Proc. IEEE*, vol. 93, no. 11, pp. 1890–1908, 2005.

[12] J. Carpentier, “Contribution to the Economic Dispatch Problem,” *Soc. Fr. Electr.*, pp. 431–447, 1962.

[13] S. Frank and S. Rebennack, “A Primer on Optimal Power Flow: Theory, Formulation, and Practical Examples,” *Color. Sch. Mines, Tech. Rep.*, no. October, pp. 1–42, 2012.

[14] New England ISO, “Maintaining Operating Reserve.” [Online]. Available: https://www.iso-ne.com/about/what-we-do/three-roles/operating-grid.

[15] J. H. Chow, R. W. De Mello, and K. W. Cheung, “Electricity market design: An integrated approach to reliability assurance,” *Proc. IEEE*, vol. 93, no. 11, pp. 1956–1968, 2005.

[16] E. Ela, M. Milligan, and B. Kirby, “Operating Reserves and Variable Generation,” *Natl. Renew. Energy Lab*, pp. 1–103, 2011.

[17] Y. Tang, K. Dvijotham, and S. Low, “Real-Time Optimal Power Flow,” *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

[18] E. Mohagheghi, M. Alramlawi, A. Gabash, and P. Li, “A survey of real-time optimal power flow,” *Energies*, vol. 11, no. 11, 2018.

[19] E. K. Richard P. O’Neill, Thomas Dautel, “Recent ISO Software Enhancements and Future

Software and Modeling Plans," *Fed. Energy Regul. Comm. Tech. Rep.*, pp. 1–42, 2011.

[20] National Academies of Sciences Engineering and Medicine, "Analytic Research Foundations for the Next-Generation Electric Grid," *Natl. Acad. Press*, pp. 1–148, 2016.

[21] B. Eldridge, R. P. O'Neill, and A. Castillo, "Marginal Loss Calculations for the DCOPF," *FERC Tech. Rep. Loss Estim.*, pp. 1–30, 2017.

[22] A. G. Bakirtzis and P. N. Biskas, "A decentralized solution to the DC-OPF of interconnected power systems," *IEEE Trans. Power Syst.*, 2003.

[23] R. C. Burchett, H. H. Happ, D. R. Vierath, and K. A. Wirgau, "Developments in Optimal Power Flow," *IEEE Trans. Power Appar. Syst.*, vol. PAS-101, no. 2, pp. 406–414, 1982.

[24] Y. Qi, D. Shi, and D. Tylavsky, "Impact of assumptions on DC power flow model accuracy," *2012 North Am. Power Symp. NAPS 2012*, no. 1, pp. 1–6, 2012.

[25] S. H. Low, "Convex relaxation of optimal power flow - Part i: Formulations and equivalence," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 15–27, 2014.

[26] K. Dvijotham and D. K. Molzahn, "Error bounds on the DC power flow approximation: A convex relaxation approach," *IEEE 55th Conf. Decis. Control. CDC 2016*, pp. 2411–2418, 2016.

[27] S. Frank and S. Rebennack, "An introduction to optimal power flow: Theory, formulation, and examples," *IIE Trans. (Institute Ind. Eng.*, vol. 48, no. 12, pp. 1172–1197, 2016.

[28] B. Stott and O. Alsaç, "Optimal power flow–basic requirements for real-life problems and their solutions," *SEPOPE XII Symp. Rio Janeiro, Brazil*, pp. 1–19, 2012.

[29] T. Heidel, F. Pan, S. Elbert, C. Demarco, and H. Mittelmann, "Optimal Power Flow Competition Design Considerations," *Fed. Energy Regul. Comm. Tech. Conf.*, pp. 1–21, 2016.

[30] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: A bibliographic survey I Formulations and deterministic methods," *Energy Syst.*, vol. 3, no. 3, pp. 221–258, 2012.

[31] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: A bibliographic survey II Non-deterministic and hybrid methods," *Energy Syst.*, vol. 3, no. 3, pp. 259–289, 2012.

[32] A. Castillo and R. P. O. Neill, "Survey of Approaches To Solving the ACOPF(OPF Paper 4)," *US Fed. Energy Regul. Comm. Staff Tech. Rep.*, pp. 1–49, 2013.

[33] F. Capitanescu, "Critical review of recent advances and further developments needed in AC optimal power flow," *Electr. Power Syst. Res.*, vol. 136, pp. 57–68, 2016.

[34] M. B. Maskar, A. R. Thorat, and I. Korachgaon, "A review on optimal power flow problem and solution methodologies," *2017 Int. Conf. Data Manag. Anal. Innov.*, pp. 64–70, 2017.

[35] F. Zohrizadeh, C. Josz, M. Jin, R. Madani, J. Lavaei, and S. Sojoudi, "A survey on conic relaxations of optimal power flow problem," *Eur. J. Oper. Res.*, pp. 1–19, 2020.

[36] IEEE Working Group on Modern Heuristic Optimization, "OPF Test Bed," 2014. [Online]. Available: https://site.ieee.org/psace-mho/.

[37] U.S. Department of Energy, "Grid Optimization Competition," 2014. [Online]. Available: https://gocompetition.energy.gov/.

[38] S. K. Mukherjee, A. Recio, and C. Douligeris, "Optimal Power Flow by Linear Programming based Optimization," *Proc. IEEE Southeastcon '92*, pp. 527–529, 1992.

[39] IEEE Power Engineering Society, "IEEE tutorial on optimal power flow: solution

techniques, requirements and challenges," *IEEE*, pp. 1–74, 1996.

[40] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, "Nonlinear Programming: Theory and Algorithms," *Wiley Intersci.*, pp. 163–256, 2006.

[41] H. W. Dommel and W. F. Tinney, "Optimal Power Flow Solutions," *IEEE Trans. Power Appar. Syst.*, vol. PAS-87, no. 10, pp. 1866–1876, 1968.

[42] X. Wang, Y. Song, and M. Irving, "Modern Power Systems Analysis," *Springer Sci. Bus. Media*, pp. 202–217, 2008.

[43] A. Castillo and R. P. O'Neill, "Computational performance of solution techniques applied to the ACOPF (OPF Paper 5)," *US Fed. Energy Regul. Comm. Staff Tech. Rep.*, pp. 15–19, 2013.

[44] N. S. Rau, *Optimization principles: Practical applications to the operation and markets of the electric power industry*. 2003.

[45] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite programming for optimal power flow problems," *Int. J. Electr. Power Energy Syst.*, vol. 30, no. 6–7, pp. 383–392, 2008.

[46] R. A. Jabr, "Optimal power flow using an extended conic quadratic formulation," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 1000–1008, 2008.

[47] Z. Yang, H. Zhong, Q. Xia, and C. Kang, "Fundamental Review of the OPF Problem: Challenges, Solutions, and State-of-the-Art Algorithms," *J. Energy Eng.*, vol. 144, no. 1, pp. 1–12, 2018.

[48] D. K. Molzahn and I. A. Hiskens, "A Survey of Relaxations and Approximations of the Power Flow Equations," *Found. Trends® Electr. Energy Syst.*, vol. 4, no. 1–2, pp. 1–221, 2019.

[49] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 92–107, 2012.

[50] D. K. Molzahn, B. C. Lesieutre, and C. L. Demarco, "A sufficient condition for global optimality of solutions to the optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 29, no. 2, pp. 978–979, 2014.

[51] S. H. Low, "Convex relaxation of optimal power flow-part II: Exactness," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 177–189, 2014.

[52] C. W. Tan, D. W. H. Cai, and X. Lou, "Resistive network optimal power flow: Uniqueness and algorithms," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 263–273, 2015.

[53] L. Gan, N. Li, U. Topcu, and S. H. Low, "Exact Convex Relaxation of Optimal Power Flow in Radial Networks," *IEEE Trans. Automat. Contr.*, vol. 60, no. 1, pp. 72–87, 2015.

[54] S. Huang, Q. Wu, J. Wang, and H. Zhao, "A Sufficient Condition on Convex Relaxation of AC Optimal Power Flow in Distribution Networks," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1359–1368, 2017.

[55] J. Li, F. Liu, Z. Wang, S. H. Low, and S. Mei, "Optimal Power Flow in Stand-Alone DC Microgrids," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5496–5506, 2018.

[56] B. C. Lesieutre, D. K. Molzahn, A. R. Borden, and C. L. DeMarco, "Examining the limits of the application of semidefinite programming to power flow problems," *2011 49th Annu. Allert. Conf. Commun. Control. Comput. Allert. 2011*, pp. 1492–1499, 2011.

[57] W. A. Bukhsh, A. Grothey, K. I. M. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4780–4788,

2013.

[58] B. Kocuk, S. S. Dey, and X. A. Sun, “Inexactness of SDP relaxation and valid inequalities for optimal power flow,” *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 642–651, 2016.

[59] Z. Yuan and M. R. Hesamzadeh, “Second-order cone AC optimal power flow: convex relaxations and feasible solutions,” *J. Mod. Power Syst. Clean Energy*, vol. 7, no. 2, pp. 268–280, 2019.

[60] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco, “Implementation of a large-scale optimal power flow solver based on semidefinite programming,” *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 3987–3998, 2013.

[61] J. Rahul, Y. Sharma, and D. Birla, “A New Attempt to Optimize Optimal Power Flow based Transmission Losses Using Genetic Algorithm,” *Proc. - 4th Int. Conf. Comput. Intell. Commun. Networks*, pp. 566–570, 2012.

[62] U. Leeton, D. Uthitsunthorn, U. Kwannetr, N. Sinsuphun, and T. Kulworawanichpong, “Power Loss Minimization Using Optimal Power Flow based on Particle Swarm Optimization,” *Electr. Eng. Comput. Telecommun. Inf. Technol. ECTICON 2010 Int. Conf.*, pp. 440–444, 2010.

[63] P. Jangir, S. A. Parmar, I. N. Trivedi, and R. H. Bhesdadiya, “A novel hybrid Particle Swarm Optimizer with multi verse optimizer for global numerical optimization and Optimal Reactive Power Dispatch problem,” *Eng. Sci. Technol. an Int. J.*, vol. 20, no. 2, pp. 570–586, 2017.

[64] T. Sousa, J. Soares, Z. A. Vale, H. Morais, and P. Faria, “Simulated Annealing Metaheuristic to solve the Optimal Power Flow,” in *IEEE Power and Energy Society General Meeting*, 2011.

[65] W. Bai, I. Eke, and K. Y. Lee, “An improved artificial bee colony optimization algorithm based on orthogonal learning for optimal power flow problem,” *Control Eng. Pract.*, vol. 61, no. February, pp. 163–172, 2017.

[66] W. S. Sakr, R. A. EL-Sehiemy, and A. M. Azmy, “Adaptive differential evolution algorithm for efficient reactive power management,” *Appl. Soft Comput. J.*, vol. 53, pp. 336–351, 2017.

[67] S. Surender Reddy and P. R. Bijwe, “Differential evolution-based efficient multi-objective optimal power flow,” *Neural Comput. Appl.*, vol. 31, pp. 509–522, 2019.

[68] S. S. Reddy, “Optimal power flow using hybrid differential evolution and harmony search algorithm,” *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 5, pp. 1077–1091, 2019.

[69] C. G. Marcelino *et al.*, “Solving security constrained optimal power flow problems: a hybrid evolutionary approach,” *Appl. Intell.*, vol. 48, no. 10, pp. 3672–3690, 2018.

[70] H. R. E. H. Bouchekara, M. A. Abido, and M. Boucherma, “Optimal power flow using Teaching-Learning-Based Optimization technique,” *Electr. Power Syst. Res.*, vol. 114, pp. 49–59, 2014.

[71] V. Roberge, M. Tarbouchi, and F. Okou, “Optimal power flow based on parallel metaheuristics for graphics processing units,” *Electr. Power Syst. Res.*, vol. 140, pp. 344–353, 2016.

[72] F. Lezama, J. Soares, Z. Vale, J. Rueda, S. Rivera, and I. Elrich, “2017 IEEE competition on modern heuristic optimizers for smart grid operation: Testbeds and results,” *Swarm Evol. Comput.*, vol. 44, no. April 2018, pp. 420–427, 2019.

[73] Y. Jin, “Surrogate-assisted evolutionary computation: Recent advances and future

challenges," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 61–70, 2011.

[74] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black - Box Functions," *J. Glob. Optim.*, vol. 13, no. 4, pp. 455–492, 1998.

[75] D. Lim, Y. Jin, Y. S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 329–355, 2010.

[76] B. Liu, Q. Zhang, and G. G. E. Gielen, "A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, 2014.

[77] R. Jin, W. Chen, and T. W. Simpson, "Comparative Studies of Metamodelling Techniques Under Multiple Modelling Criteria," *Struct. Multidiscip. Optim.*, vol. 23, no. 1, pp. 1–13, 2001.

[78] A. Díaz-Manríquez, G. Toscano, and C. A. Coello Coello, "Comparison of Metamodeling Techniques in Evolutionary Algorithms," *Soft Comput.*, vol. 21, no. 19, pp. 5647–5663, 2017.

[79] S. Xiao, "Balancing exploration and exploitation in robust multiobjective electromagnetic design optimisation," The University of Southampton, 2014.

[80] Y. Li, "Robust Multiobjective Optimization in Electromagnetic Design," The University of Southampton, 2017.

[81] A. H. R. de M. B. Kranz, "Real-Time Scheduling: Real-Time Commitment(RTC) and Real-Time Dispatch(RTD)," *Prelim. Des. Considerations*, 2002.

[82] NYISO, "Manual 11: Day-Ahead Scheduling Manual (Version 4.6)," *NYISO Energy Mark. Oper.*, pp. 1–82, 2017.

[83] PJM, "PJM Manual 11: Scheduling Operations," 2005. [Online]. Available: https://pjm.com/training/certification/~/media/training/nerc-certifications/m11v26.ashx.

[84] H. Chen, "Security Constrained Economic Dispatch System (SCED) Overview," 2016. [Online]. Available: https://www.aeso.ca/assets/Uploads/3.3-SCED-Overview-by-PJM.pdf.

[85] CAISO, "Real-Time Market," *Fifth Replace. Electron. Tarif.*, pp. 1–46, 2016.

[86] M. Huneault and F. D. Galiana, "A Survey Of The Optimal Power Flow Literature," *IEEE Trans. Power Syst.*, 1991.

[87] FERC Joint Boards on Security Constrained Economic Dispatch, "Study and Recommendations Regarding Security Constrained Economic Dispatch (SCED) in the Northeast by the Joint Board on Economic Dispatch for the Northeast Region," *FERC Docket AD05-13*, pp. 14–15, 2006.

[88] A. Castillo, S. Member, C. Laird, C. A. Silva-monroy, J. Watson, and R. P. O. Neill, "The Unit Commitment Problem With AC Optimal Power Flow Constraints," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4853–4866, 2016.

[89] M. S. Osman, M. A. Abo-Sinna, and A. A. Mousa, "A solution to the optimal power flow using genetic algorithm," *Appl. Math. Comput.*, vol. 155, no. 2, pp. 391–405, 2004.

[90] O. Alsaç, J. Bright, M. Prais, and B. Stott, "Further developments in lp-based optimal power flow," *IEEE Trans. Power Syst.*, 1990.

[91] A.-A. A. Mohamed, Y. S. Mohamed, A. A. M. El-Gaafary, and A. M. Hemeida, "Optimal power flow using moth swarm algorithm," *Electr. Power Syst. Res.*, vol. 142, pp. 190–206, 2017.

[92] L. R. De Araujo, D. R. R. Penido, and F. D. A. Vieira, “A multiphase optimal power flow algorithm for unbalanced distribution systems,” *Int. J. Electr. Power Energy Syst.*, 2013.

[93] T. Sousa, H. Morais, Z. Vale, and R. Castro, “A multi-objective optimization of the active and reactive resource scheduling at a distribution level in a smart grid context,” *Energy*, 2015.

[94] X.-P. Zhang, “Fundamentals of electric power systems,” *IEEE Restructured Electr. Power Syst. Anal. Electr. Mark. with Equilib. Model.*, pp. 1–52, 2010.

[95] A. Soroudi, “Power system optimization modeling in GAMS,” *Switz. Springer*, pp. 1–6, 2017.

[96] H. Abdi, S. D. Beigvand, and M. La Scala, “A review of optimal power flow studies applied to smart grids and microgrids,” *Renewable and Sustainable Energy Reviews*. 2017.

[97] R. Shigenobu, A. Yona, and T. Senjyu, “Optimal demand response considering the optimal power flow in electricity market,” *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2016-May, pp. 523–528, 2016.

[98] A. Venzke and S. Chatzivasileiadis, “Convex Relaxations of Probabilistic AC Optimal Power Flow for Interconnected AC and HVDC Grids,” *IEEE Trans. Power Syst.*, 2019.

[99] A. T. Azevedo, A. R. L. Oliveira, M. J. Rider, and S. Soares, “How to efficiently incorporate facts devices in optimal active power flow model,” *J. Ind. Manag. Optim.*, 2010.

[100] Z. Yang, H. Zhong, Q. Xia, and C. Kang, “Optimal Transmission Switching with Short-Circuit Current Limitation Constraints,” *IEEE Trans. Power Syst.*, 2016.

[101] L. W. De Oliveira, F. D. S. Seta, and E. J. De Oliveira, “Optimal reconfiguration of distribution systems with representation of uncertainties through interval analysis,” *Int. J. Electr. Power Energy Syst.*, 2016.

[102] J. A. Momoh *et al.*, “Challenges to optimal power flow,” *IEEE Power Eng. Rev.*, vol. 17, no. 2, pp. 71–72, 1997.

[103] B. A. H. W. F. Tineey, J. M. Bright, K. D. Demaree, “Some deficiencies in optimal power flow,” *IEEE Trans. Power Syst.*, vol. 3, no. 2, pp. 676–683, 1988.

[104] F. Capitanescu *et al.*, “State-of-the-art, challenges, and future trends in security constrained optimal power flow,” *Electr. Power Syst. Res.*, vol. 81, no. 8, pp. 1731–1741, 2011.

[105] Z. Yang, H. Zhong, Q. Xia, and C. Kang, “A novel network model for optimal power flow with reactive power and network losses,” *Electr. Power Syst. Res.*, vol. 144, pp. 63–71, 2017.

[106] D. K. Molzahn, “Computing the Feasible Spaces of Optimal Power Flow Problems,” *IEEE Trans. Power Syst.*, vol. 32, no. 6, pp. 4752–4763, 2017.

[107] B. Stott, J. Jardim, and O. Alsaç, “DC power flow revisited,” *IEEE Trans. Power Syst.*, 2009.

[108] A. Garces, “On the convergence of Newton’s method in power flow studies for dc microgrids,” *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5770–5777, 2018.

[109] X. L. Jianzhe Liu, Bai Cui, Daniel K. Molzahn, Chen Chen, “Optimal Power Flow for DC Networks with Robust Feasibility and Stability Guarantees,” *arXiv Prepr. arXiv1902.08163*, pp. 1–9, 2019.

[110] A. Castillo, P. Lipka, J. P. Watson, S. S. Oren, and R. P. O’Neill, “A Successive Linear

Programming Approach to Solving the IV-ACOPF," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 2752–2763, 2016.

[111] K. Purchala, L. Meeus, D. Van Dommelen, and R. Belmans, "Usefulness of DC power flow for active power flow analysis," *2005 IEEE Power Eng. Soc. Gen. Meet.*, vol. 1, pp. 454–459, 2005.

[112] M. Ilić, S. Cvijić, J. H. Lang, J. Tong, and D. Obadina, "Operating beyond today's PV curves: Challenges and potential benefits," *IEEE Power Energy Soc. Gen. Meet.*, vol. 2015-Septe, 2015.

[113] Y. Xu, J. Ma, Z. Y. Dong, and D. J. Hill, "Robust Transient Stability-Constrained Optimal Power Flow with Uncertain Dynamic Loads," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1911–1921, 2016.

[114] J. Gong, D. Xie, C. Jiang, and Y. Zhang, "A new solution for stochastic optimal power flow: Combining limit relaxation with iterative learning control," *J. Electr. Eng. Technol.*, vol. 9, no. 1, pp. 80–89, 2014.

[115] H. Zhang and P. Li, "Probabilistic analysis for optimal power flow under uncertainty," *IET Gener. Transm. Distrib.*, vol. 4, no. 5, pp. 553–561, 2010.

[116] H. Glavitsch and M. Spoerry, "Quadratic loss formula for reactive dispatch," *IEEE Trans. Power Appar. Syst.*, vol. PAS-102, no. 12, pp. 3850–3858, 1983.

[117] M. R. AlRashidi and M. E. El-Hawary, "Hybrid particle swarm optimization approach for solving the discrete OPF problem considering the valve loading effects," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 2030–2038, 2007.

[118] R. A. Jabr, "Radial distribution load flow using conic programming," *IEEE Trans. Power Syst.*, vol. 21, no. 3, pp. 1458–1459, 2006.

[119] G. C. Contaxis, C. Delkis, and G. Korres, "Decoupled optimal load flow using linear or quadratic programming," *IEEE Trans. Power Syst.*, 1986.

[120] I. R. Virginijus Radziukynas, "Optimization methods application to optimal power flow in electric power systems," *Optim. Energy Ind.*, pp. 409–436, 2009.

[121] X. Xia and A. M. Elaiw, "Optimal dynamic economic dispatch of generation: A review," *Electr. power Syst. Res.*, vol. 80, no. 8, pp. 975–986, 2010.

[122] S. Boyd and L. Vandenberghe, *Convex Optimization*. 2004.

[123] G. J. M. Victor Klee, "How good is the simplex algorithm?," *Washingt. Univ.*, pp. 159–175, 1970.

[124] J. Nocedal and W. Stephen, "Numerical Optimization," *Springer Sci. Bus. Media*, pp. 388–389, 2006.

[125] R. Robere, "Interior Point Methods and Linear Programming," *Univ. Toronto*, pp. 1–15, 2012.

[126] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Proc. Sixt. Annu. ACM Symp. Theory Comput.*, vol. 4, pp. 373–395, 1984.

[127] R. Fourer, "Solving Linear Programs by Interior-Point Method," *Dep. Ind. Eng. Manag. Sci. Northwest. Univ.*, pp. B73–B94, 2004.

[128] M. Kojima, S. Mizuno, and A. Yoshise, "A Primal-Dual Interior Point Algorithm for Linear Programming," *Prog. Math. Program.*, pp. 1–19, 1989.

[129] G. Tzallas-regas, “Primal-Dual Interior Point algorithms for Linear Programming Linear Programming and Optimality Con- ditions,” pp. 1–8, 1984.

[130] O. Güler, D. den Hertog, C. Roos, T. Terlaky, and T. Tsuchiya, “Degeneracy in interior point methods for linear programming: a survey,” *Ann. Oper. Res.*, vol. 46, pp. 107–138, 1993.

[131] and M. E. L. Gauthier, Jean Bertrand, Jacques Desrosiers, “Tools for primal degenerate linear programs :,” *EURO J. Transp. Logist.*, vol. 5, no. 2, pp. 1–39, 2016.

[132] N. S. Rau, “Issues in the path toward an RTO and standard markets,” *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 435–443, 2003.

[133] J. Peschon, D. W. Bree, and L. P. Hajdu, “Optimal Power-Flow Solutions for Power System Planning,” *Proc. IEEE*, vol. 60, no. 1, pp. 64–70, 1972.

[134] G. Strang, “Calculus,” *Massachusetts Inst. Technol. Wellesley-Cambridge Press*, pp. 91–164, 1991.

[135] D. I. Sun, B. Ashley, B. Brewer, A. Hughes, and W. F. Tinney, “Optimal Power Flow By Newton Approach,” *IEEE Trans. Power Appar. Syst.*, vol. PAS-103, no. 10, pp. 2864–2880, 1984.

[136] G. N. Vanderplaats, “Numerical Optimization Techniques for Engineering Design,” *Vanderplaats Research & Development*. pp. 84–121, 1999.

[137] F. Capitanescu and L. Wehenkel, “Experience With the Multiple Centrality Corrections Interior Point Algorithm for Optimal Power Flow,” *CEE Conf. Coimbra, Port.*, no. 1, pp. 1–6, 2005.

[138] A. V Fiacco and G. P. McCormick, “The Sequential Unconstrained Minimization Technique for Nonlinear Programing , a Primal- Dual Method,” *Manage. Sci.*, vol. 10, no. 2, pp. 360–366, 1964.

[139] W. Hua, H. Sasaki, K. Junji, and Y. Ryuichi, “An interior point nonlinear programming for optimal power flow problems with a novel data structure,” *Proc. 20th Int. Conf. Power Ind. Comput. Appl.*, pp. 134–141, 1997.

[140] M. Salahi, J. Peng, and T. Terlaky, “On Mehrotra-Type Predictor-Corrector Algorithms,” *SIAM J. Optim.*, vol. 18, no. 4, pp. 1377–1397, 2005.

[141] F. Capitanescu and L. Wehenkel, “Experiments with the interior-point method for solving large scale Optimal Power Flow problems,” *Electr. Power Syst. Res.*, vol. 95, pp. 276–283, 2013.

[142] D. K. Molzahn, B. C. Lesieutre, and C. L. DeMarco, “Investigation of non-zero duality gap solutions to a semidefinite relaxation of the optimal power flow problem,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 2325–2334, 2014.

[143] D. K. Molzahn and I. A. Hiskens, “Sparsity-Exploiting Moment-Based Relaxations of the Optimal Power Flow Problem,” *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3168–3180, 2015.

[144] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, “Strengthening the SDP Relaxation of AC Power Flows with Convex Envelopes, Bound Tightening, and Valid Inequalities,” *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3549–3558, 2017.

[145] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, “Applications of second-order cone programming,” *Linear Algebra Appl.*, vol. 284, no. 1–3, pp. 193–228, 1998.

[146] R. A. Jabr, “A conic quadratic format for the load flow equations of meshed networks,”

IEEE Trans. Power Syst., vol. 22, no. 4, pp. 2285–2286, 2007.

[147] M. Farivar and S. H. Low, "Branch flow model: Relaxations and convexification-part i," *IEEE Trans. Power Syst.*, vol. 28, no. 3, pp. 2554–2564, 2013.

[148] B. Kocuk, S. S. Dey, and X. Andy Sun, "Strong SOCP relaxations for the optimal power flow problem," *Oper. Res.*, vol. 64, no. 6, pp. 1177–1196, 2016.

[149] R. R. T. Blekherman, Grigoriy, Pablo A. Parrilo, "Semidefinite Optimization and Convex Algebraic Geometry," *Soc. Ind. Appl. Math.*, pp. 3–40, 2012.

[150] R. M. Freund, "Introduction to Semidefinite Programming (SDP)," *Massachusetts Inst. Technol.*, pp. 1–54, 2004.

[151] X. Wei and H. Bai, "Semi-definite programming-based method for security-constrained unit commitment with operational and optimal power flow constraints," *Gener. Transm. Distrib. IET*, vol. 1, no. 2, pp. 182–197, 2007.

[152] B. Zhang and D. Tse, "Geometry of injection regions of power networks," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 788–797, 2013.

[153] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 199–211, 2015.

[154] H. Mahboubi and J. Lavaei, "Analysis of Semidefinite Programming Relaxation of Optimal Power Flow for Cyclic Networks," *Proc. IEEE Conf. Decis. Control*, pp. 3203–3210, 2018.

[155] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC Relaxation: A Theoretical and Computational Study on Optimal Power Flow," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 3008–3018, 2016.

[156] S. Sojoudi and J. Lavaei, "On the exactness of semidefinite relaxation for nonlinear optimization over graphs: Part II," *Proc. IEEE Conf. Decis. Control*, pp. 1051–1057, 2013.

[157] C. Bingane, M. F. Anjos, and S. Le Digabel, "Tight-and-cheap conic relaxation for the AC optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 7181–7188, 2018.

[158] M. NIU, C. WAN, and Z. XU, "A review on applications of heuristic optimization algorithms for optimal power flow in modern power systems," *J. Mod. Power Syst. Clean Energy*, vol. 2, no. 4, pp. 289–297, 2014.

[159] M. K. N. Patel, "A Review of Artificial Intelligence Based Optimization Techniques for Optimal Power Flow," *Int. J. Adv. Eng. Res. Dev.*, vol. 3, no. 1, pp. 105–112, 2016.

[160] J. Soares, M. A. Fotouhi Ghazvini, M. Silva, and Z. Vale, "Multi-dimensional signaling method for population-based metaheuristics: Solving the large-scale scheduling problem in smart grids," *Swarm Evol. Comput.*, vol. 29, pp. 13–32, 2016.

[161] F. Lezama, L. E. Sucar, E. M. De Cote, J. Soares, and Z. Vale, "Differential evolution strategies for large-scale energy resource management in smart grids," *GECCO 2017 - Proc. Genet. Evol. Comput. Conf. Companion*, pp. 1279–1286, 2017.

[162] R. Malhotra, N. Singh, and Y. Singh, "Genetic Algorithms: Concepts, Design for Optimization of Process Controllers," *Comput. Inf. Sci.*, vol. 4, no. 2, pp. 39–54, 2011.

[163] S. Mirjalili, J. Song Dong, and A. Lewis, "Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction," *Nature-Inspired Optim.*, pp. 73–89, 2020.

[164] F. Vafaee, G. Turán, P. C. Nelson, and T. Y. Berger-Wolf, "Balancing the exploration and exploitation in an adaptive diversity guided genetic algorithm," *Proc. 2014 IEEE Congr.*

Evol. Comput. CEC 2014, pp. 2570–2577, 2014.

[165] Y. Y. Wong, K. H. Lee, K. S. Leung, and C. W. Ho, "A novel approach in parameter adaptation and diversity maintenance for genetic algorithms," *Soft Comput.*, 2003.

[166] T. Baeck, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," *IEEE Conf. Evol. Comput. - Proc.*, pp. 57–62, 1994.

[167] A. R. Ayad, H. A. Awad, and A. A. Yassin, "Parametric analysis for genetic algorithms handling parameters," *Alexandria Eng. J.*, vol. 52, no. 1, pp. 99–111, 2013.

[168] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," *IEEE Trans. Syst. Man Cybern.*, vol. 24, no. 4, pp. 656–667, 1994.

[169] Q. H. Wu, Y. J. Cao, and J. Y. Wen, "Optimal reactive power dispatch using an adaptive genetic algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 20, no. 8, pp. 563–569, 1998.

[170] A. F. Attia, Y. A. Al-Turki, and A. M. Abusorrah, "Optimal power flow using adapted genetic algorithm with adjusting population size," *Electr. Power Components Syst.*, vol. 40, no. 11, pp. 1285–1299, 2012.

[171] A. L. H. Mills, Kevin L., James J. Filliben, "Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters," *Evol. Comput.*, vol. 23, no. 2, pp. 309–342, 2015.

[172] M. Crepinsek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, 2013.

[173] A. Hussain and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," *Complex Intell. Syst.*, vol. 6, no. 1, pp. 1–14, 2020.

[174] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. Int. Conf. Neural Networks*, pp. 1942–1948, 1995.

[175] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *IEEE Int. Conf. Evol. Comput. Proceedings. IEEE World Congr. Comput. Intell.*, pp. 69–73, 1998.

[176] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, 2002.

[177] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive Particle Swarm Optimization," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 6, pp. 1362–1381, 2009.

[178] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," *3rd World Congr. Nat. Biol. Inspired Comput.*, pp. 633–640, 2011.

[179] R. Eberhart and J. Kennedy, "A New Optimizer using Particle Swarm Theory," in *Proceedings of the International Symposium on Micro Machine and Human Science*, 1995.

[180] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," *IEEE Swarm Intell. Symp.*, pp. 120–127, 2007.

[181] A. P. Engelbrecht, "Particle Swarm Optimization: Global Best or Local Best?," *Proc. - 11th BRICS Ctries. Congr. Comput. Intell. BRICS-CCI 2013*, pp. 124–135, 2013.

[182] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[183] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," *Proc. 1999 Congr. Evol. Comput. CEC 1999*, vol. 3, pp. 1931–1938, 1999.

[184] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," *Proc. 2002 Congr. Evol. Comput. CEC 2002*, vol. 2, pp. 1671–1676, 2002.

[185] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, 2004.

[186] R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 36, no. 4, pp. 515–519, 2006.

[187] W. H. Lim and N. A. Mat Isa, "Adaptive division of labor particle swarm optimization," *Expert Syst. Appl.*, vol. 42, no. 14, pp. 5887–5903, 2015.

[188] M. Vora and T. T. Mirnalinee, "Dynamic small world particle swarm optimizer for function optimization," *Nat. Comput.*, vol. 17, no. 4, pp. 901–917, 2018.

[189] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 3, pp. 627–646, 2012.

[190] W. H. Lim and N. A. Mat Isa, "An adaptive two-layer particle swarm optimization with elitist learning strategy," *Inf. Sci. (Ny).*, vol. 273, pp. 49–72, 2014.

[191] L. Zhang, J. Sun, C. Guo, and H. Zhang, "A Multi-swarm Competitive Algorithm Based on Dynamic Task Allocation Particle Swarm Optimization," *Arab. J. Sci. Eng.*, vol. 43, no. 12, pp. 8255–8274, 2018.

[192] M. Younes, M. Rahli, and L. Abdelhakem-Koridak, "Optimal Power Flow based on Hybrid Genetic Algorithm," *J. Inf. Sci. Eng.*, vol. 23, no. 6, pp. 1801–1816, 2007.

[193] L. Shengsong, W. Min, and H. Zhijian, "Hybrid algorithm of chaos optimisation and SLP for optimal power flow problems with multimodal characteristic," *IEEE Proceedings-Generation Transm. Distrib.*, vol. 150, no. 5, pp. 543–547, 2003.

[194] J. Chuanwen and E. Bompard, "A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation," *Math. Comput. Simul.*, vol. 68, no. 1, pp. 57–65, 2005.

[195] S. Surender Reddy, P. R. Bijwe, and A. R. Abhyankar, "Faster evolutionary algorithm based optimal power flow using incremental variables," *Int. J. Electr. Power Energy Syst.*, vol. 54, pp. 198–210, 2014.

[196] Z. Yang, A. Bose, H. Zhong, N. Zhang, Q. Xia, and C. Kang, "Optimal Reactive Power Dispatch with Accurately Modeled Discrete Control Devices: A Successive Linear Approximation Approach," *IEEE Trans. Power Syst.*, vol. 32, no. 3, pp. 2435–2444, 2017.

[197] S. Surender Reddy and P. R. Bijwe, "Day-Ahead and Real Time Optimal Power Flow considering Renewable Energy Resources," *Int. J. Electr. Power Energy Syst.*, vol. 82, pp. 400–408, 2016.

[198] J. G. Robertson, G. P. Harrison, and A. R. Wallace, "OPF Techniques for Real-Time Active Management of Distribution Networks," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3529–3537, 2017.

[199] P. P. Vergara, R. Torquato, and L. C. P. da Silva, "Towards a real-time Energy Management System for a Microgrid using a multi-objective genetic algorithm," *IEEE Power Energy Soc. Gen. Meet.*, pp. 1–5, 2015.

[200] S. Surender Reddy, P. R. Bijwe, and A. R. Abhyankar, “Optimal posturing in day-ahead market clearing for uncertainties considering anticipated real-time adjustment costs,” *IEEE Syst. J.*, vol. 9, no. 1, pp. 177–190, 2015.

[201] T. Niknam, M. R. Narimani, J. Aghaei, S. Tabatabaei, and M. Nayeripour, “Modified Honey Bee Mating Optimisation to solve dynamic optimal power flow considering generator constraints,” *IET Gener. Transm. Distrib.*, vol. 5, no. 10, pp. 989–1002, 2011.

[202] R. F. Gunst, R. H. Myers, and D. C. Montgomery, “Building Empirical Models,” *Response Surf. Methodol. Process Prod. Optim. Using Des. Exp. John Wiley Sons*, pp. 13–80, 1996.

[203] K. P. Murphy, “Machine learning: a probabilistic perspective (adaptive computation and machine learning series),” *MIT Press Cambridge, Massachusetts London, Engl.*, pp. 219–246, 2012.

[204] M. Laguna and R. Martí, “Experimental testing of advanced scatter search designs for global optimization of multimodal functions,” *J. Glob. Optim.*, vol. 33, no. 2, pp. 235–255, 2005.

[205] N. Dyn, D. Levin, and S. Rippa, “Numerical Procedures for Surface Fitting of Scattered Data by Radial Functions,” *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 2, pp. 639–659, 1986.

[206] M. J. D. Powell, “Radial basis functions for multivariable interpolation: a review,” *Algorithms Approx.*, pp. 143–167, 1987.

[207] H. M. Gutmann, “A Radial Basis Function Method for Global Optimization,” *J. Glob. Optim.*, vol. 19, no. 3, pp. 201–227, 2001.

[208] J. S. Søren N. Lophaven, Hans Bruun Nielsen, “DACE - A Matlab Kriging Toolbox [Software],” *Informatics and Mathematical Modelling, Technical University of Denmark, DTU*, 2002. [Online]. Available: http://www2.imm.dtu.dk/pubdb/pubs/1460-full.html.

[209] C. Kuan, “Introduction to Econometric Theory,” pp. 39–108, 2000.

[210] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, “Design and Analysis of Computer Experiments,” *Stat. Sci.*, vol. 4, no. 4, pp. 409–423, 1989.

[211] D. R. Jones, “A Taxonomy of Global Optimization Methods Based on Response Surfaces,” *J. Glob. Optim.*, vol. 21, no. 4, pp. 345–383, 2001.

[212] H. P. W. Jerome Sacks, William J. Welch, Toby J. Mitchell, “Design and Analysis of Computer Experiments,” *Stat. Sci.*, vol. 4, no. 4, pp. 409–435, 1989.

[213] M. Schonlau, W. J. Welch, and D. R. Jones, “Global versus local search in constrained optimization of computer models,” *New Dev. Appl. Exp. Des.*, pp. 11–25, 1998.

[214] A. Sóbester, S. J. Leary, and A. J. Keane, “On the design of optimization strategies based on global response surface approximation models,” *J. Glob. Optim.*, vol. 33, no. 1, pp. 31–59, 2005.

[215] Y. Li, S. Xiao, M. Rotaru, and J. K. Sykulski, “A Kriging-Based Optimization Approach for Large Data Sets Exploiting Points Aggregation Techniques,” *IEEE Trans. Magn.*, vol. 53, no. 6, pp. 1–4, 2017.

[216] Y. Li, S. Xiao, M. Rotaru, and J. K. Sykulski, “A Dual Kriging Approach with Improved Points Selection Algorithm for Memory Efficient Surrogate Optimization in Electromagnetics,” *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, 2016.

[217] Y. Tenne and S. W. Armfield, “A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications,” *Success Comput. Intell.*, pp. 43–72, 2008.

[218] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 1, pp. 66–76, 2007.

[219] B. S. Jin, Yaochu, "Fitness approximation in evolutionary computation - A survey," *4th Annu. Conf. Genet. Evol. Comput.*, pp. 1105–1112, 2002.

[220] Y. Jin, "A Comprehensive Survey of Fitness Approximation in Evolutionary Computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, 2005.

[221] A. Díaz-Manríquez, G. Toscano, J. H. Barron-Zambrano, and E. Tello-Leal, "A Review of Surrogate Assisted Multiobjective Evolutionary Algorithms," *Comput. Intell. Neurosci.*, pp. 1–14, 2016.

[222] J. J. Grefenstette and J. M. Fitzpatrick, "Genetic search with approximate fitness evaluations," *1st Int. Conf. Genet. Algorithms*, pp. 112–120, 1985.

[223] D. E. Grierson and W. H. Pak, "Optimal sizing, geometrical and topological design using a genetic algorithm," *Struct. Optim.*, vol. 6, no. 3, pp. 151–159, 1993.

[224] A. Ratle, "Accelerating the convergence of evolutionary algorithms by fitness landscape approximation," *Int. Conf. Parallel Probl. Solving from Nat.*, pp. 87–96, 1998.

[225] A. Ratle, "Optimal Sampling Strategies for Learning a Fitness Model," *Congr. Evol. Comput.*, pp. 2078–2085, 1999.

[226] L. Bull, "On Model-Based Evolutionary Computation," *Soft Comput. - A Fusion Found. Methodol. Appl.*, vol. 3, no. 2, pp. 76–82, 1999.

[227] Y. Jin, M. Olhofer, and B. Sendhoff, "On Evolutionary Optimization with Approximate Fitness Functions," *2nd Annu. Conf. Genet. Evol. Comput.*, pp. 786–793, 2000.

[228] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, 2002.

[229] K. Rasheed and H. Hirsh, "Informed operators: Speeding up Genetic-algorithm-based design Optimization using Reduced Models," *2nd Annu. Conf. Genet. Evol. Comput.*, pp. 628–635, 2000.

[230] K. S. Anderson and Y. Hsu, "Genetic crossover strategy using an approximation concept," *Congr. Evol. Comput.*, vol. 1, pp. 527–533, 1999.

[231] M. S. Abboud, K., "Surrogate deterministic mutation: Preliminary results," *Int. Conf. Artif. Evol.*, pp. 104–116, 2001.

[232] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft Comput.*, vol. 19, no. 6, pp. 1461–1475, 2015.

[233] H. Yu, Y. Tan, J. Zeng, C. Sun, and Y. Jin, "Surrogate-assisted hierarchical particle swarm optimization," *Inf. Sci. (Ny).*, vol. 454, pp. 59–72, 2018.

[234] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[235] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," pp. 100–105, 1994.

[236] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, 1998.

[237] D. Whitley, “A Genetic Algorithm Tutorial,” *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.

[238] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “A Comparative Study of Different Variants of Genetic Algorithms for Constrained Optimization,” *Asia-Pacific Conf. Simulated Evol. Learn.*, pp. 177–186, 2010.

[239] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 57–73, 2002.

[240] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, “Adaptive particle swarm optimization.,” *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 39, no. 6, pp. 1362–1381, 20099.

[241] T. Bäck and H.-P. Schwefel, “An Overview of Evolutionary Algorithms for Parameter Optimization,” *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, 1993.

[242] S. Duman, “A modified moth swarm algorithm based on an arithmetic crossover for constrained optimization and optimal power flow problems,” *IEEE Access*, vol. 6, pp. 45394–45416, 2018.

[243] B. Stott, O. Alsac, and A. J. Monticelli, “Security Analysis and Optimization,” *Proc. IEEE*, vol. 72, no. 12, pp. 1623–1644, 1987.

[244] R. D. Zimmerman and C. E. Murillo-Sánchez, “MATPOWER 6.0 [Software],” 2016. [Online]. Available: https://matpower.org.

[245] a. Bhattacharya and P. K. Chattopadhyay, “Application of biogeography-based optimisation to solve different optimal power flow problems,” *IET Gener. Transm. Distrib.*, vol. 5, no. 1, pp. 70–80, 2011.

[246] S. Surender Reddy and C. Srinivasa Rathnam, “Optimal Power Flow using Glowworm Swarm Optimization,” *Int. J. Electr. Power Energy Syst.*, vol. 80, pp. 128–139, 2016.

[247] A. Ananthi Christy and P. A. D. Vimal Raj, “Adaptive biogeography based predator-prey optimization technique for optimal power flow,” *Int. J. Electr. Power Energy Syst.*, vol. 62, pp. 344–352, 2014.

[248] F. van den Bergh and A. P. Engelbrecht, “A Cooperative Approach to Particle Swarm Optimization,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, 2004.

[249] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, “Orthogonal Learning Particle Swarm Optimization,” *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, 2011.

[250] Y. W. Leung, Y. Wang, Y. W. Leung, and Y. Wang, “An Orthogonal Genetic Algorithm with Quantization for Global Numerical Optimization,” *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 41–53, 2001.

[251] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, 2006.

[252] I. Erlich, K. Y. Lee, J. L. Rueda, and S. Wildenhues, “Competition on Application of Modern Heuristic Optimization Algorithms for Solving Optimal Power Flow Problems Problem Definitions and Implementation Guidelines,” pp. 1–11, 2014.

[253] Z. Deng, M. D. Rotaru, and J. K. Sykulski, “A study of evolutionary based optimal power flow techniques,” *51st Int. Univ. Power Eng. Conf.*, pp. 1–6, 2016.

[254] IEEE Working Group on Modern Heuristic Optimization, “Competition on Application of Modern Heuristic Optimization Algorithms for Solving Optimal Power Flow Problems,” 2014. [Online]. Available: https://site.ieee.org/psace-mho/panels-and-competitions-2014-

opf-problems/.

[255] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, “Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction,” *Struct. Multidiscip. Optim.*, vol. 53, no. 5, pp. 935–952, 2016.

[256] K. Deb and R. B. Agrawal, “Simulated Binary Crossover for Continuous Search Space,” *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

Appendix A

A1. Test the Performance of Polynomial Regression

The following Matlab codes can be directly implemented for building 2nd-order PR model with 11 samples, as plotted in Fig. 3.1 (a). The parameters "*sz*" and "*pd*" can be used to define the sample size and polynomial degree to build the desirable PR models.

```
%% Test Polynomial Regression
% x := predicting points;    y := predicting points' real values; y_hat := predicted values;
% S := samples points;    Y := samples' objective function values;
% pd := polynomial degree to be set;    sz := step size;
% min_y_hat :=  predicted minimum value;    min_x_hat :=  the point of which has minimum predicted value;
% min_y :=  minimum real objective value;    min_x :=  the point of which has real minimum value;
% id_p := the index of minimum predicted value;    id_r := the index of real minimum value;
x = -500:500; % with step size 1
y = zeros(1,size(x,2));
for i = 1 : size(x,2)
    y(i) = Schwefel (x(i));
end
%% samples
sz = 100;
S = -500 : sz : 500; % with step size 100
Y = zeros(1,size(S,2));
for i = 1 : size(S,2)
    Y(i) = Schwefel (S(i));
end
% normalize the data for better curve fitting
nx = (x-mean(x))./std(x);    ny = (y-mean(y))./std(y);
nS = (S-mean(S))./std(S);    nY = (Y-mean(Y))./std(Y);
%% Polynomial evaluation
pd = 3;
beta = polyfit (nS,nY,pd);  % find the polynomial coefficient
y_hat = polyval(beta,nx); % Polynomial curve fitting
y_hat = y_hat.*std(Y) + mean(Y); % restore to the original data
%% find the predicted minimum
[min_y_hat,id_p] = min(y_hat);
min_x_hat = x(id_p);
%% find the real minimum
[min_y,id_r] = min(y); min_x = x(id_r);
%% plot
plot(x,y,'LineWidth',2);
xlabel('x'); ylabel('y');
hold on
plot(x,y_hat,'LineWidth',1.5)
scatter(S,Y,20,'black','filled');
scatter(min_x_hat,min_y_hat,65,'red','filled');
scatter(min_x,min_y,35,'green','filled');
legend('Schwefel Function','PR Model','Samples','Estimated Minimum','Real Minimum')
title('2nd PR model with 11 samples')
```

The "***Schwefel***" function is coded as follows

```
function f = Schwefel(x)
%% The Schwefel Function
    d = length(x); % number of dimensions
    f = 0;
    for ii = 1:d
        xi = x(ii); f = f + xi*sin(sqrt(abs(xi)));
    end
    f = 418.9829*d - f;
end
```

A2. Test the Performance of Radial Basis Function

The following Matlab codes can be directly implemented for building cubic RBF model with 51 samples, as plotted in Fig. 3.2 (e). The parameters "*sz*" and "*basis*" can be used to define the sample size and basis function to build the preferable RBF models.

```
%% Test Radial Basis Function
% x := predicting points;    y := predicting points' real values; y_hat := predicted values;
% S := samples points;    Y := samples' objective function values;
% sz := step size;    basis := selected basis function
% min_y_hat :=  predicted minimum value;    min_x_hat :=  the point of which has minimum predicted value;
% min_y :=  minimum real objective value;    min_x :=  the point of which has real minimum value;
% id_p := the index of minimum predicted value;    id_r := the index of real minimum value;


x = -500:500;
y = zeros(1,size(x,2));

for i = 1 : size(x,2)

    y(i) = Schwefel(x(i));

end

%% sampling
sz = 20;
S = -500 : sz : 500;
Y = zeros(1,size(S,2));

for i = 1 : size(S,2)

    Y(i) = Schwefel(S(i));

end

%% RBF curve fitting
basis = 'cubic'; % or 'gauss' 'spline', 'multiquadric', or 'linear'
y_hat = RBF(S,Y,x,basis);

%% find the predicted minimum
[min_y_hat,id_p] = min(y_hat);
min_x_hat = x(id_p);

%% find the real minimum
[min_y,id_r] = min(y);
min_x = x(id_r);

%% plot
plot(x,y,'LineWidth',2);
xlabel('x');
ylabel('y');
hold on
plot(x,y_hat,'LineWidth',1.5)
scatter(S,Y,20,'black','filled');
scatter(min_x_hat,min_y_hat,65,'red','filled');
scatter(min_x,min_y,35,'green','filled');
legend('Schwefel Function','RBF Model','Samples','Estimated Minimum','Real Minimum')
title('Cubic RBF model with 51 samples')
```

Note the above "***RBF***" function is not an inner Matlab function, rather it is coded as follows:

```
function y_hat = RBF(S,Y,x,basis)

%% RBF
% S := samples points;    Y := samples' objective function values;    x := predicting points;
% y_hat := predicted values;

    S = S'; x = x'; Y = Y';
    [m,~] = size(S); % sample size
    [mx,~] = size(x);

    %% normlizing data
    S_mean = mean(S);
    S_std = std(S);
    Y_mean = mean(Y);
    Y_std = std(Y);
    nS = (S-S_mean)./S_std;
    nY = (Y-Y_mean)./Y_std;
    nx = (x-S_mean)./S_std;

    %% find the distances
    r_s = pdist2(nS,nS,'squaredeuclidean'); % pairwise distance between each sample
    r_sx = pdist2(nx,nS,'squaredeuclidean'); % pairwise distance between samples and estimations

    %% find the coefficients for the RBF model using different basis function
    if basis == "gauss"
        theta = 1;
        phi_s = exp(-theta.*r_s);
        phi_sx = exp(-theta.*r_sx);

    elseif basis == "cubic"
        phi_s = r_s.^3;
        phi_sx = r_sx.^3;

    elseif basis == "spline"
        phi_s = r_s.^2.*log(r_s);
        phi_sx = r_sx.^2.*log(r_sx);

    elseif basis == "multiquadric"
        theta = 5;
        phi_s = (r_s.^2+theta^2);
        phi_sx = (r_sx.^2+theta^2);

    elseif basis == "linear"
        phi_s = r_s;
        phi_sx = r_sx;

    else
        return;
    end

    phi_s(logical(eye(m,m))) = 1+(10*m)*eps;  % this is important to avoid singular or near singular matrix
    phi_sx(isnan(phi_sx))=0;
    phi_sx = reshape(phi_sx,mx,m);

    %% predictor
    lambda = phi_s\nY; % find the coefficients for the radial basis function model
    y_hat = phi_sx*lambda; % output estimations
    y_hat = y_hat.*Y_std + Y_mean; % restore the data


end
```

A3. Kriging Predictor

The Matlab codes of simple and efficient "***kriging_predictor***" used in this thesis and modified from Kriging DACE toolbox are given as follows:

```
function [y_hat,MSE] = kriging_predictor(S,Y,x,theta)

    %% kriging fitting
    % S := samples points;    Y := samples' objective function values;    x := predicting points;
    % y_hat := predicted values; MSE := potential prediction errors
    [m, ~] = size(S); [mx, ~] = size(x);
    %% Normalize data
    mS = mean(S);   sS = std(S);
    mY = mean(Y);   sY = std(Y);
    S = (S - mS) ./ sS;
    x = (x - mS) ./ sS;
    Y = (Y - mY) ./ sY;

    %% correlation matrix
    % pairwise correlation between each sample
    S = S.*sqrt(theta);
    D = pdist2(S,S,'squaredeuclidean')';
    R = exp(-D);
    R(logical(eye(m,m))) = 1 + (10+m)*eps; % to avoid singular matrix

    % pairwise correlation between sample and estimation
    x = x.*sqrt(theta);
    dx = (pdist2(S,x,'squaredeuclidean'));
    r = exp(-dx);

    %% Regression methods
    % this is like in the PR, here we use linear PR, while other polynomial
    % degree can be used
    F = ones(m,1);
    f = ones(mx,1);

    % Get least squares solution using QR decomposition
    % not the F and Y cannot directly used for practical computation if the
    % problem is large or ill-conditioned. So the GLS solutions should be found
    % by orthogonal transformation as follows
    [C, ~] = chol(R); % Chol decomposition
    C = C';
    Ft = C \ F; Yt = C \ Y;
    [Q, G] = qr(Ft,0);
    beta = G \ (Q'*Yt); % GLS optimal mean value
    rho = Yt - Ft*beta;
    gamma = rho' / C; % equal to the term R^(-1)*(y-xu) in equation (3.21)
    sigma2 = sum(rho.^2)/m; % optimal variance
    sigma2 = sY.^2.*sigma2; % restore to  original value

    %% kriging predictor

    ny = f * beta + (gamma * r).';

    y_hat = mY + sY .* ny; % restore to  original value

    %% MSE
    rt = C \ r;
    u = G' \ (Ft.' * rt - f.');
    MSE = sigma2 .* (1 + sum(u.^2,1) - sum(rt.^2))';

end
```

A4. Kriging Optimization using Estimated Minimum Criterion

The Matlab codes of using Kriging and Estimated minimum Criterion to find the global optimum are given as below:

```
%% Kriging optimization using estimated minimum
clear all
close all
% x := predicting points;  y := predicting points' real values; y_hat := % predicted values;
% S := samples points;    Y := samples' objective function values;
% sz := step size; min_id := the index of minimum point; MaxIt := maximum iterations
% min_y_hat :=  estimated minimum value;    min_x :=  estimated minimum point;
% inf_y := infill point's objective value;    inf_x :=  selected infill point;

% for plotting the original Schwefel function with step size 1
x = -500:500;
y = zeros(1,size(x,2));
for i = 1 : size(x,2)
    y(i) = Schwefel(x(i));
end
x = x';
y = y';
% samples
sz = 200;
S = -500 : sz : 500;
Y = zeros(1,size(S,2));

for i = 1 : size(S,2)
    Y(i) = Schwefel(S(i));
end
S = S';
Y = Y';
MaxIt = 8;
theta = 5;
for it = 1 : MaxIt
 %% Kriging Estimation and iteratively selecting the infill point
    [y_hat,~] = kriging_predictor(S,Y,x,theta); % Kriging predictor
% find the estimated minimum
    [min_y_hat,min_id] = min(y_hat);
    min_x_hat = x(min_id);
% using estimated minimum as infill point
    inf_x = min_x_hat;
    inf_y = Schwefel(inf_x);
% update the sample set as well as the Kriging model
    S(end+1) = inf_x;
    Y(end+1) = inf_y;
    [min_y,min_id] = min(y);
    min_x = x(min_id);
%% plot
    figure(it)
    plot(x,y,'LineWidth',2);
    xlabel('x');
    ylabel('y');
    hold on
    plot(x,y_hat,'LineWidth',1.5)
    scatter(S(1:end-1),Y(1:end-1),20,'black','filled');
    scatter(min_x,min_y,35,'green','filled');
    scatter(min_x_hat,min_y_hat,35,'red');
    scatter(S(end),Y(end),45,'m','*');
    legend('Schwefel Function','Kriging Model','Samples','Real Minimum','Estimated Minimum','Infilling Point')
    title(['Iteration ' num2str(it)] )
    hold off
end
```

A5. Kriging Optimization using Maximum MSE Criterion

The Matlab codes of using Kriging and Maximum MSE Criterion to find the global optimum are given as below:

```
%% Kriging optimization using maximum MSE
% x := predicting points;  y := predicting points' real values; y_hat := % predicted values;
% S := samples points;    Y := samples' objective function values;
% sz := step size; max_id := the index of maximum MSE; MaxIt := maximum iterations
% min_y_hat :=  estimated minimum value;    min_x :=  estimated minimum point;
% inf_y := infill point's objective value;    inf_x :=  selected infill point;
% max_mse := value of maximum MSE; max_mse_x := the point of which has maximum MSE


x = -500:500;
y = zeros(1,size(x,2));
for i = 1 : size(x,2)
    y(i) = Schwefel(x(i));
end
x = x';  y = y';
% samples
sz = 200; S = -500 : sz : 500;  Y = zeros(1,size(S,2));

for i = 1 : size(S,2)
    Y(i) = Schwefel(S(i));
end
S = S'; Y = Y';
MaxIt = 8;
theta = 5;
for it = 1 : MaxIt

%% Kriging Estimation and iteratively selecting the infill point
    [y_hat,MSE] = kriging_predictor(S,Y,x,theta); % Kriging predictor
    MSE = normalize(MSE,'range'); % normalizing MSE
    [min_y_hat,min_id] = min(y_hat); min_x_hat = x(min_id);
% find the point of which has maximum MSE
    [max_mse,max_id] = max(MSE);  max_mse_x = x(max_id);
% using the point of which has maximum MSE as the infill point
    inf_x = max_mse_x; inf_y = Schwefel(inf_x);
% update the sample set as well as the Kriging model
    S(end+1) = inf_x;
    Y(end+1) = inf_y;
    [min_y,max_id] = min(y); min_x = x(max_id);

%% plot
    figure(it)
    yyaxis left
    plot(x,y,'LineWidth',2);
    hold on
    plot(x,y_hat,'-','LineWidth',1.5,'Color',[0.9290 0.6940 0.1250])
    scatter(S(1:end-1),Y(1:end-1),20,'black','filled');
    scatter(min_x,min_y,20,'green','filled');
    scatter(min_x_hat,min_y_hat,30,'red');
    scatter(S(end),Y(end),35,'m','*');
    xlabel('x'); ylabel('y');
    yyaxis right
    plot(x,MSE,'--','Color',[0.8500 0.3250 0.0980],'LineWidth',1)
    ylim([0,10]);
    ylabel('MSE');
    scatter(max_mse_x,max_mse,35,'c','filled');
    yyaxis left
    legend('Schwefel Function','Kriging Model','Samples','Real Minimum','Estimated Minimum','Infilling
    title(['Iteration ' num2str(it)] )
    hold off
end
```

A6. Kriging Optimization using Maximum Weighted Expected Improvement

The following Matlab codes can be used to test the Kriging using MWEI criterion with different weigh value (by changing the parameter "*weight_ei*") on Schwefel and Griewank functions:

```
%% Kriging optimization using maximum EI
% x := predicting points;  y := predicting points' real values; y_hat := % predicted values;
% S := samples points;    Y := samples' objective function values;
% sz := step size; max_id := the index of maximum EI; MaxIt := maximum iterations
% min_y_hat :=  estimated minimum value;    min_x :=  estimated minimum point;
% inf_y := infill point's objective value;    inf_x :=  selected infill point;
% max_ei := value of maximum EI; max_ei_x := the point of which has maximum EI
% weight_ei := the weight defined for calculating Expected Improvement
% for Griewank, x = -10:10; sz = 3; S = -10:sz:10;
x = -500:500;  y = zeros(1,size(x,2));
for i = 1 : size(x,2)
    y(i) = Schwefel(x(i));
end
x = x'; y = y';
% samples
sz = 200; S = -500 : sz : 500; Y = zeros(1,size(S,2));
for i = 1 : size(S,2)
    Y(i) = Schwefel(S(i));
end
S = S'; Y = Y';
MaxIt = 20; weight_ei = 0.5;
theta = 5;
for it = 1 : MaxIt
%% Kriging Estimation and iteratively selecting the infill point
    [y_hat,MSE] = kriging_predictor(S,Y,x,theta); % Kriging predictor

    [EI,max_EI_idx]=  WMEI(MSE,y_hat,Y,weight_ei); % find the maximum EI
    EI = normalize(EI,'range'); % normalizing EI
% find point of which has maximum EI
    max_EI_x = x(max_EI_idx);
% using the point of which has maximum EI as the infill point
    inf_x = max_EI_x; inf_y = Schwefel(inf_x);
% update the sample set as well as the Kriging model
    S(end+1) = inf_x;  Y(end+1) = inf_y;
    [min_y,id] = min(y);  min_x = x(id);
%% plot
    figure(it)
    yyaxis left
    plot(x,y,'LineWidth',2);
    hold on
    plot(x,y_hat,'-','LineWidth',1.5,'Color',[0.9290 0.6940 0.1250])
    scatter(S(1:end-1),Y(1:end-1),20,'black','filled');
    scatter(min_x,min_y,20,'green','filled');
    scatter(S(end),Y(end),35,'m','*');
    xlabel('x');
    ylabel('y');
    yyaxis right
    plot(x,EI,'--','Color',[0.8500 0.3250 0.0980],'LineWidth',1)
    ylim([0,10]);
    ylabel('EI');
    scatter(max_EI_x,EI(max_EI_idx),35,'c','filled');
    yyaxis left
    legend('Schwefel Function','Kriging Model','Samples','Real Minimum','Infilling Point','EI','Maximum EI')
    title(['Iteration ' num2str(it)] )
    hold off
end
```

Note the parameter "*weight_ei*" can be used to favor the exploitative and explorative search power, and the "***WMEI***" function used above to calculate the Weighted Maximum Expected Improvement is given as below

```
function [EI,max_EI_idx]= WMEI(MSE,y_hat,Y,weight)
%% Calculation of weighted Expected Improvement
% weight := defined weight for EI
% MSE := prediction errors
% Y_min : = current minimum solution
% y_hat : = estimations
% max_EI_idx: = the index of which point has maximum EI value
% EI := the calculated EI

    sd = sqrt(max(0,MSE)); % standard deviation

    Y_min = min(Y);

    [m, ~] = size(y_hat);

    % calculate the EI using equation (3.27)
    EI = weight.*(Y_min-y_hat).*normcdf((Y_min-y_hat)./sd) + (1-weight).*sd.*normpdf((Y_min-y_hat)./sd);

    % if MSE is equal to zero, the corrsponding EI value is set to zero
    for i = 1:m

        if MSE(i,1)==0

            EI(i,1) = 0;

        end

    end

    % find the weighted maximum EI
    [~,max_EI_idx] = max(EI);

end
```

Moreover, the "***Griewank***" function is coded as follows:

```
function f = Griewank (x)
%% the Griewank Function
    d = length(x); % number of variables
    f = 0;
    prod = 1;
    for ii = 1:d
        xi = x(ii);
        f = f + xi^2/4000;
        prod = prod * cos(xi/sqrt(ii));
    end
    f = f - prod + 1;
end
```

A7. Kriging Efficient Global Optimization

The following Matlab codes are the EGO algorithm, which can use IPM, PSO or other preferred solvers to maximize MWEI to find the optimal solutions of different optimization problems.

```
function [t,sol] = KEGO
%% the Kriging Efficient Global Optimization
    lb = -500; ub = 500; % for Schwefel function test
    %lb = -600; ub = 600; % for Griewank function test
    nVar = 5; % number of variables
    MaxIt = 500; % max iteration
    alg = 1; % alg = 1 means KEGO-IPM, alg = 2 means KEGO-PSO;
    ss = 50; % sample size
    S = zeros(ss,nVar); Y = zeros(ss,1); theta = 5;
    for i = 1:ss
        S(i,:) = unifrnd(lb,ub,1,nVar);  Y(i,1) = Schwefel(S(i,:)); % generate uniformly random initial samples
    end
    if alg == 1
        IPM_pa = optimoptions('fmincon','Algorithm','interior-point',...
            'MaxIterations',20,'Display','off'); % set the parameters for the IPM
    elseif alg == 2
        PSO_pa = optimoptions('particleswarm','SwarmSize',10,...
            'MaxIterations',100,'Display','off'); % set the parameters for the PSO
    end
    sol = zeros(1,500);  t = zeros(1,500);
    for it = 1:MaxIt
        tic
        if alg == 1
            myfun=@(x) ei_obj_func(x,S,Y,theta); % the objective function for finding the MWEI
        % use IPM to find the optimal infill point based on MWEI
            [opt_infill_x, ~] = fmincon(myfun,unifrnd(-500,500,1,nVar),[],[],...
                [],[],repmat(lb,1,nVar),repmat(ub,1,nVar),[],IPM_pa); % use IPM
        elseif alg == 2
            % use PSO to find the optimal infill point based on MWEI
            [opt_infill_x , ~] = particleswarm(myfun,nVar,repmat(lb,1,nVar),...
                repmat(ub,1,nVar),PSO_pa); % use PSO
        end
        % update the sample
        S = [S; opt_infill_x];  Y = [Y; Schwefel(opt_infill_x)];
        t(it) = toc; % recorde the CPU time at iteration "it"
        sol(it) = min(Y); % recorde the best solution found so far
        fprintf('Number of function evaluation: %4.0f; Best feasible function value: %f\n', size(S,1),min(Y))
    end
end
```

The above "*ei_obj_func*" is the MWEI objective function, which is coded as follow

```
function opt_EI = ei_obj_func(x,S,Y,theta)
%% the objective function to find the maximum weighted expected improvement value
    [y_hat,MSE] = kriging_predictor(S,Y,x,theta); % predict the value at point x
    weight = 0.5; % set the weight for WMEI
    [EI,~]= WMEI(MSE,y_hat,Y,weight); % calculate the EI at point x
    % due to the MATLAB IPM and PSO solvers devote to find the minimum value, but the solvers
    % used in EGO should be aimed at finding the maximum value. Therefore the negative sign is used to EI.
    opt_EI = -EI; % output objective EI value.
end
```

Here the "*ei_obj_func*" function coded in this simple form is for illustration purpose. To improve the computational efficiency of EOG, above "*ei_obj_func*" function could be further improved by pre-computing the Kriging parameters "*beta*", "*gamma*" and "*sigma2*" and thus the "*ei_obj_func*" function only needs to mainly compute the "*y_hat*", "*MSE*" and "*EI*", which is much more efficient.

A8. Kriging Assisted Genetic Algorithm

The following Matlab codes of KAGA can be directly used to minimize the generation cost of the IEEE 118 bus test system. To solve the OPF problems, this coded Matlab script should be placed in the same directory with MATPOWER 6.0 (i.e. a power flow analysis tool). By changing the "*test_case*", it can be switched to solve the numerical optimization problems, while specifying "*test_fucntion*" to solve different numerical functions. Moreover, "*opf_system*" and "*opf_obj*" can be used to define different OPF test systems and objective functions, respectively. Furthermore, the "***KAGA***" can be switched back to the original GA by setting "*test_alg*" to 0.

```
function [final_fitness, lambda] = KAGA
    addpath([cd '/matpower6.0/']);
    rng('default')
    %% test case preparation
    global nVar nPop VarMax VarMin theta sigma it MaxIt test_case test_function opf_system opf_obj
    % chose the test algorithm
    test_alg = 1; % 0 means original GA, 1 means Kriging assisted GA
    % chose the test cases
    test_case = 2; % 1 means numerical function tests, 2 means OPF tests
    if test_case == 1
    % select the numerical functions, e.g. 'Ackley', 'Griewank',' Levy', 'Rastrigin', 'Schwefel'
        test_function = 'Ackley';
        [~,VarMin,VarMax] = numerical_Obj_Func([],test_function);
    elseif test_case == 2
    % select the test systems and the OPF objective functions
        opf_system = 'ieee_118_bus'; % or 'ieee_118_bus'
        opf_obj = 'cost'; % or 'losses'
        [VarMin,VarMax,mpc] = OPF_Initialization(opf_system);
    end
    % VarMin := Lower Bound of Variables, VarMax := Upper Bound of Variables
    %% GA parameters
    nVar = length(VarMin); % Number of control Variables
    VarSize = [1 nVar]; % Size of Decision Variables Matrix
    if test_case == 1
        MaxIt = 50000; % Maximum Number of Iterations
        nPop = 20; % Population Size (Swarm Size)
    elseif test_case == 2
        switch opf_system
            case 'ieee_30_bus'
                if test_alg == 0
                    MaxIt = 100; % Maximum Number of Iterations
                elseif test_alg == 1
                    MaxIt = 500; % Maximum Number of Iterations
                end
                nPop = 50; % Population Size (Swarm Size)
            case 'ieee_118_bus'
                if test_alg == 0
                    MaxIt = 1000; % Maximum Number of Iterations
                elseif test_alg == 1
                    MaxIt = 10000; % Maximum Number of Iterations
                end
                nPop = 100; % Population Size (Swarm Size)
        end
    end
    pc = 0.7;  % Crossover Percentage
    nc = 2*round(pc*nPop/2); % Number of Offspring (also Parents)
    pm = 0.3; % Mutation Percentage
    nm = round(pm*nPop); % Number of Mutants
    mu = 0.01; % Mutation Rate
    %% Kriging and ELS parameters, tuning them can achieve better results
    if test_case == 1
```

```
        theta = 0.05;
        sigma = 1;
    elseif test_case == 2
        switch opf_system
            case 'ieee_30_bus'
                theta = 0.001;
            case 'ieee_118_bus'
                theta = 0.0035;
        end
        sigma = 0.1;
    end
    %% Initialization
    S = zeros(nPop,nVar);
    Y = zeros(nPop,1);
    empty_individual.Position=[];
    empty_individual.Cost=[];
    pop=repmat(empty_individual,nPop,1);
    for i = 1:nPop
        % Initialize Position
        pop(i).Position = unifrnd(VarMin,VarMax,VarSize);
        % Fitness Evaluation
        if test_case == 1 % numerical case
            [pop(i).Cost,~,~] = numerical_Obj_Func(pop(i).Position,test_function);
        elseif test_case == 2 % opf test
            [pop(i).Cost, mpc] = OPF_Obj_Func(pop(i).Position,mpc);
        end
        % Update Samples
        S(i,:) = pop(i).Position;
        Y(i,1) = pop(i).Cost;
    end
    % Sort Population
    Costs = [pop.Cost];
    [~, SortOrder] = sort(Costs);
    pop = pop(SortOrder);
    % record current global best
    GlobalBest.Cost = pop(1).Cost;
    GlobalBest.Position = pop(1).Position;
    %% main optimization loop
    for it = 1 : MaxIt
    %% Crossover
        popc=repmat(empty_individual,nc/2,2);
        for k = 1:nc/2
            % Roulette Wheel Selection
            i1 = Selection([pop.Cost]); i2 = Selection([pop.Cost]);
            % Select Parents
            p1 = pop(i1); p2 = pop(i2);
            % Apply Crossover
            [popc(k,1).Position, popc(k,2).Position] = Crossover(p1.Position,p2.Position);
            if test_alg == 0
            % Evaluate Offspring
                if test_case == 1 % numerical case
                    [popc(k,1).Cost,~,~] = numerical_Obj_Func(popc(k,1).Position,test_function);
                    [popc(k,2).Cost,~,~] = numerical_Obj_Func(popc(k,2).Position,test_function);
                elseif test_case == 2 % opf case
                    [popc(k,1).Cost,mpc] = OPF_Obj_Func(popc(k,1).Position,mpc);
                    [popc(k,2).Cost,mpc] = OPF_Obj_Func(popc(k,2).Position,mpc);
                end
            end
        end
        popc = popc(:);
    %% Mutation
        popm = repmat(empty_individual,nm,1);
        for k = 1:nm
            % Randomly Select an offspring to mutation
            i = randi([1 nc]); p = popc(i);
            % Apply Mutation
            popm(k).Position = Mutation(p.Position,mu);
```

```
            if test_alg == 0
            % Evaluate mutation
                if test_case == 1 % numerical test
                    [popm(k).Cost,~,~] = numerical_Obj_Func(popm(k).Position,test_function);
                elseif test_case == 2 % opf test
                    [popm(k).Cost,mpc] = OPF_Obj_Func(popm(k).Position,mpc);
                end
            end
        end
        offpsring = [popc;popm]; % offspring data
    %% perform the Individual based Kriging Strategy
        if  test_alg == 1
            x = reshape([offpsring.Position],nVar,nm+nc)'; % reshape the structure data "offspring"
            if test_case == 1
                [S,Y,pop,pop(1),~] = Individual_Kriging(S,Y,x,pop,pop(1)); % numerical test
            elseif test_case == 2
                [S,Y,pop,pop(1),mpc] = Individual_Kriging(S,Y,x,pop,pop(1),mpc); % opf test
            end
        end
    %% perform the Elitist Learning Strategy
        if test_case == 1 % numerical test
            [S,Y,pop,pop(1)] = ELS(S,Y,pop,pop(1));
        elseif test_case == 2 % opf test
            [S,Y,pop,pop(1),mpc] = ELS(S,Y,pop,pop(1),mpc);
        end
    %% Sort Population
        if test_alg == 0 % for original GA only
            all_pop = [pop;offpsring]; % store the old population and offspring together
            Costs = [all_pop.Cost];
            [~, SortOrder] = sort(Costs);
            pop = all_pop(SortOrder(1:nPop)); % only the nPop best individuals survival
        elseif test_alg == 1
            Costs = [pop.Cost];
            [~, SortOrder] = sort(Costs);
            pop = pop(SortOrder);
        end
    %% record current global best
        GlobalBest.Cost = pop(1).Cost;
        GlobalBest.Position = pop(1).Position;
    %% display the current best found in the iteration "it"
        disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(GlobalBest.Cost)]);
    end
    if test_case == 1
        final_fitness = GlobalBest.Cost;
        lambda = [];
    elseif test_case == 2
        %% use Reactive Power Enforcement strategy to deal with the Qg violations
        [final_fitness,lambda] = RPES(GlobalBest,mpc); % output the final value and penalty
    end
end
```

The "***numerical_Obj_Func***" function used in the "***KAGA***" script for defining the necessary parameters of numerical optimization functions is coded as follows:

```
function [fitness,LB,UB] = numerical_Obj_Func(x,func)
%% Numerical Benchmark Optimization Functions
%change to different objective functions
    switch func
        case 'Ackley'
            if ~isempty(x)
                fitness = Ackley(x);
                LB = []; UB = [];
            else
                LB = repmat(-32,1,50); UB = repmat(32,1,50);
                fitness = [];
            end
        case 'Griewank'
```

```
            if ~isempty(x)
                fitness = Griewank(x);
                LB = []; UB = [];
            else
                LB = repmat(-10,1,50); UB = repmat(10,1,50);
                fitness = [];
            end
        case 'Levy'
            if ~isempty(x)
                fitness = Levy(x);
                LB = []; UB = [];
            else
                LB = repmat(-10,1,50); UB = repmat(10,1,50);
                fitness = [];
            end
        case 'Rastrigin'
            if ~isempty(x)
                fitness = Rastrigin(x);
                LB = []; UB = [];
            else
                LB = repmat(-5.12,1,50); UB = repmat(5.12,1,50);
                fitness = [];
            end
        case 'Schwefel'
            if ~isempty(x)
                fitness = Schwefel(x);
                LB = []; UB = [];
            else
                LB = repmat(-500,1,50); UB = repmat(500,1,50);
                fitness = [];
            end
    end
end
```

The scripts of "***Schwefel***" and "***Griewank***" are already provided in the A1 and A6, respectively. Other "***Ackley***", "***Levy***" and "***Rastrigin***" functions are given as follows:

```
function f = Ackley(x)
%% the ACKLEY Function
    a = 20; b = 0.2; c = 2*pi;
    ff1 = 0; ff2 = 0; d = length(x);
    for ii = 1:d
        xi = x(ii); ff1 = ff1 + xi^2; ff2 = ff2 + cos(c*xi);
    end
    f1 = -a * exp(-b*sqrt(ff1/d)); f2 = -exp(ff2/d); f = f1 + f2 + a + exp(1);
end

function f = Levy(x)
%% the levy function
    d = length(x);
    for ii = 1:d
        w(ii) = 1 + (x(ii) - 1)/4;
    end
    f1 = (sin(pi*w(1)))^2; f2 = (w(d)-1)^2 * (1+(sin(2*pi*w(d)))^2); f3 = 0;
    for ii = 1:(d-1)
        wi = w(ii); new = (wi-1)^2 * (1+10*(sin(pi*wi+1))^2); f3 = f3 + new;
    end
    f = f1 + f3 + f2;
end

function f = Rastrigin(x)
%% the Rastrigin Function
    d = length(x); f = 0;
    for ii = 1:d
        xi = x(ii); f = f + (xi^2 - 10*cos(2*pi*xi));
    end
    f = 10*d + f;
```

```
end
```

The "***OPF_Initialization***" function used in "***KAGA***" script for initializing the necessary OPF parameters is coded as follows:

```
function [LB,UB,mpc] = OPF_Initialization(opf_system)
    global PD QD PG PMAX PMIN QG QMAX QMIN BS VM VA VG VMAX VMIN TAP...
        p1 p2 p3 p4 p5 p6 p7 p8 ref pv pq gen_pv vg_bus tap sh
    %% load the power system data
    % bus=:bus data, gen=:gen data, branch=:branch data
    switch opf_system
        case 'ieee_30_bus'
            mpc = case_ieee30;
        case 'ieee_118_bus'
            mpc = case118;
    end
    mpc = ext2int(mpc);
    [~, mpc.bus, mpc.gen, mpc.branch] = deal(mpc.baseMVA, mpc.bus, mpc.gen, mpc.branch);
    [ref, pv, pq] = bustypes(mpc.bus, mpc.gen); % rows of slack, PV and PQ buses in "bus"
    %% giving the index information for the power system data format
    BUS_TYPE = 2;% bus type information column in "bus"
    PD = 3; % active load columns in "bus"
    QD = 4; % reactive load columns in "bus"
    GEN_BUS = 1; % bus number column in "gen"
    PG = 2; % active power output columns in "gen"
    PMAX = 9; % maximum active power output column in "gen"
    PMIN = 10; % minimum active power output column in "gen"
    QG = 3; % reactive power output columns in "gen"
    QMAX = 4; % maximum reactive power output column in "gen"
    QMIN = 5; % minimum reactive power output column in "gen"
    BS = 6; % shunt susceptance column in "bus"
    VM = 8; % voltage magnitude columns in "bus"
    VA = 9; % voltage angle columns in "bus"
    VG = 6; % terminal voltage columns in "gen"
    VMAX = 12; % maximum voltage magnitude columns in "bus"
    VMIN = 13; % minimum voltage magnitude columns in "bus"
    TAP = 9; % tap changer columns in "bus"
    %% bus type and control variables
    gen_pv = find(mpc.gen(:,GEN_BUS) ~= ref); % find the pv bus rows in "gen"
%     gen_ref = find(gen(:,GEN_BUS) == ref); % find the slack bus row in "gen"
    vg_bus = find(mpc.bus(:,BUS_TYPE) ~= 1); % find the pv and slack buses at "bus"
    tap = find(mpc.branch(:,TAP) ~= 0); % find the taps rows in "branch"
    sh = find(mpc.bus(:,BS)~=0); % the shunts rows in "bus"
    %% number of control variables
    npg = length(pv); % total number of generators
    nvg = length(vg_bus); % total number of generator terminal voltage magnitudes
    nt = length(tap); % total number of taps
    nsh = length(sh); % total number of shunts
    %% limits of control variables
    switch opf_system
        case 'ieee_30_bus'
            Pgmax = [80;50;35;30;40]; % active outputs maximum limits
            Pgmin = [20;15;10;10;12]; % active outputs minimum limits
            Vgmax = repmat(1.1,nvg,1); % terminal voltage magnitudes maximum limits
            Vgmin = repmat(0.95,nvg,1); % terminal voltage magnitudes minimum limits
            sh = [10 12 15 17 20 21 23 24 29]';
            nsh = length(sh); % total number of shunts
            Qcmax = repmat(5,nsh,1); % shunts maximum limits
            Qcmin = zeros(nsh,1); % shunts minimum limits
        case 'ieee_118_bus'
            Pgmax = mpc.gen(gen_pv,PMAX); % active outputs maximum limits
            Pgmin = mpc.gen(gen_pv,PMIN); % active outputs minimum limits
            Vgmax = mpc.bus(vg_bus,VMAX); % terminal voltage magnitudes maximum limits
            Vgmin = mpc.bus(vg_bus,VMIN); % terminal voltage magnitudes minimum limits
            Qcmax = repmat(30,nsh,1); % shunts maximum limits
            Qcmin = zeros(nsh,1); % shunts minimum limits
    end
```

```
    Tmax = repmat(1.1,nt,1); % taps maximum limits
    Tmin = repmat(0.9,nt,1); % taps minimum limits
    %% re-organizing all control limits
    UB = [Pgmax;Vgmax;Tmax;Qcmax]'; % upper bounds
    LB = [Pgmin;Vgmin;Tmin;Qcmin]'; % lower bounds
    %% index of corrsponding variables in the variable vector
    p1 = 1; p2 = npg;  % p1 : p2 = generator outputs
    p3 = p2 + 1; p4 = p2 + nvg; % p3 : p4 = generator voltage magnitudes
    p5 = p4 + 1; p6 = p4 + nt; % p5 : p6 = taps
    p7 = p6 + 1; p8 = p6 + nsh; % p7 : p8 = shunts
end
```

The "***OPF_Obj_Func***" used in "***KAGA***" script for calculating the objective function and penalty values is programmed as follows

```
function [fitness,mpc,lambda] = OPF_Obj_Func(x,mpc)
    global PD PG QG QMAX QMIN BS VM VG TAP...
       p1 p2 p3 p4 p5 p6 p7 p8 pq gen_pv vg_bus tap sh opf_system opf_obj
    %% input the control variables to "bus","gen" and "branch" data sets
    mpc.gen(gen_pv,PG)= x(p1:p2); mpc.bus(vg_bus,VM) = x(p3:p4);
    mpc.gen(:,VG) = x(p3:p4); mpc.branch(tap,TAP) = x(p5:p6); mpc.bus(sh,BS) = x(p7:p8);
    %% run MATPOWER power flow (i.e. the default Newton Power Flow Analysis)
    [~,mpc.bus,mpc.gen,mpc.branch]=runpf(mpc);
    %% check the state variables (i.e. reactive power output Qg and load buses voltage magnitudes Vpq) feasibility
    switch opf_system
       case 'ieee_30_bus'
          Vpq_max = 1.05; % Vpq maximum
          Vpq_min = 0.95; % Vpq minimum
          penalty_Vpq = 1000; % penalty factor of reactive power output
          penalty_Qg = 1; % penalty factor of PQ buses voltage magnitudes
       case 'ieee_118_bus'
       Vpq_max = 1.06; % Vpq maximum
       Vpq_min = 0.94; % Vpq minimum
       penalty_Vpq = 5000; % penalty factor of reactive power output
       penalty_Qg = 0.02; % penalty factor of PQ buses voltage magnitudes
    end
    Qg_max = mpc.gen(:,QMAX); % Qg maximum
    Qg_min = mpc.gen(:,QMIN); % Qg minimum
    % check maximum Qg limits and calculate the penalty
    if any(mpc.bus(pq,VM) > Vpq_max)
       lambda1 = sum((mpc.bus(pq(mpc.bus(pq,VM) > Vpq_max),VM)-Vpq_max)) * penalty_Vpq;
    else
       lambda1 = 0;
    end
    % check minimum Qg limits and calculate the penalty
    if any(mpc.bus(pq,VM) < Vpq_min)
       lambda2 = sum((Vpq_min-mpc.bus(pq(mpc.bus(pq,VM) < Vpq_min),VM))) * penalty_Vpq;
    else
       lambda2 = 0;
    end
    % check maximum Vpq limits and calculate the penalty
    if any(mpc.gen(:,QG) > mpc.gen(:,QMAX))
       lambda3 = sum((mpc.gen(mpc.gen(:,QG) > Qg_max,QG) - ...
          mpc.gen(mpc.gen(:,QG) > Qg_max,QMAX))) * penalty_Qg;
    else
       lambda3 = 0;
    end
    % check minimum Vpq limits and calculate the penalty
    if any(mpc.gen(:,QG) < mpc.gen(:,QMIN))
       lambda4 = sum((mpc.gen(mpc.gen(:,QG) < Qg_min,QMIN) - ...
          mpc.gen(mpc.gen(:,QG) < Qg_min,QG))) * penalty_Qg;
    else
       lambda4 = 0;
    end
    lambda = lambda1 + lambda2 + lambda3 + lambda4; % totla penalty
    %% fitness
    switch opf_obj
```

```
        case 'cost' % minimize generation costs
            % cost coefficients
            a = mpc.gencost(:,5); b = mpc.gencost(:,6); c = mpc.gencost(:,7);
            fitness = a'*((mpc.gen(:,PG)).^2) + b'*(mpc.gen(:,PG)) + sum(c) + lambda;
        case 'losses' % minimize avtive power losses
            fitness = sum(mpc.gen(:,PG)) - sum(mpc.bus(:,PD)) + lambda;
    end
end
```

The GA operators used in the "***KAGA***" for generating offspring are coded as follows:

"***Selection***"

```
function i = Selection(Costs)
%% RouletteWheelSelection
%     r=rand;
%     p=cumsum((1./Costs)./sum(1./Costs));
%     i=find(r<=p, 1, 'first');
%% Random Selection
    global nPop
    i = randi([1 nPop]);
end
```

"***Crossover***"

```
function [y1, y2] = Crossover(x1,x2)
    global VarMin VarMax
% x1 := parent1; x2 := parent2; y1 := offspring1; y1 := offspring2
%% Single Point Crossover
%     % randomly generate the crossover point
%     c = randi([1 nVar-1]);
%     % crossover and exchange information between parents
%     y1 = [x1(1:c) x2(c+1:end)]; y2 = [x2(1:c) x1(c+1:end)];
%% Arithmetic Crossover
    gamma = 0.5;
    alpha = unifrnd(-gamma,1+gamma,size(x1));
    y1 = alpha.*x1+(1-alpha).*x2; y2 = alpha.*x2+(1-alpha).*x1;
    % enforce the limits
    y1 = max(y1,VarMin); y1 = min(y1,VarMax);
    y2 = max(y2,VarMin); y2 = min(y2,VarMax);
end
```

"***Mutation***"

```
function y = Mutation(x,mu)
    global nVar VarMin VarMax test_case
% x := the selected offspring to be mutated; y := mutated offspring
%% Non-Uniform Mutation
%     nmu = ceil(mu*nVar);
%     j = randsample(nVar,nmu)';  y = x;
%     b = 1; % the larger the b the larger the (1-rand^((1-it/MaxIt)^b)) tend to be
%     if rand > 0.5
%         y(j) = x(j)+0.1*(VarMax(j)-VarMin(j)).*(1-rand^((1-(it/MaxIt))^b));
%     else
%         y(j) = x(j)-0.1*(VarMax(j)-VarMin(j)).*(1-rand^((1-(it/MaxIt))^b));
%     end
% % enforce limits
%     y = max(y,VarMin);     y = min(y,VarMax);
%% Gaussian Mutation
    if test_case == 1
        range = 0.1;
    elseif test_case == 2
        range = 0.6;
    end
    nVar = numel(x);
    nmu = ceil(mu*nVar);
    j = randsample(nVar,nmu)';
```

```
    sigma = range*(VarMax-VarMin);
    y = x;
    y(j) = x(j)+sigma(j).*randn(size(j));
  % enforce limits
    y = max(y,VarMin);
    y = min(y,VarMax);
  end
```

"***ELS***"

```
  function [S,Y,PBest,GlobalBest,mpc] = ELS(S,Y,PBest,GlobalBest,mpc)
  %% Elitist Learning Strategy(ELS)
    global it MaxIt nVar VarMax VarMin sigma test_case test_function
    % sigma define the variance of the ELS normal distribution
    if test_case == 1
       sigma = 1 - 0.9*it/MaxIt;
    end
    d = randsample(nVar,1); % randomly choose a dimension to be mutation
    mutated_x = GlobalBest.Position; % chose the global best for mutation
    % ELS gaussian mutation
    mutated_x(d) = mutated_x(d) + (VarMax(d) - VarMin(d)) * normrnd(0,sigma);
    % enforce the limits
    mutated_x = max(mutated_x,VarMin);
    mutated_x = min(mutated_x,VarMax);
    % evalute the mutation
    if test_case == 1
       [mutated_fitness,~,~] = numerical_Obj_Func(mutated_x,test_function);
       mpc = [];
    elseif test_case == 2
       [mutated_fitness,mpc] = OPF_Obj_Func(mutated_x,mpc);
    end
    % better-adoption and worse-abandonment" strategy for updating samples
    % and memory swarm (for pso) or population (for GA)
    if mutated_fitness <= GlobalBest.Cost
       GlobalBest.Cost=mutated_fitness;
       GlobalBest.Position=mutated_x;
       [~,min_id] = min(Y);
       Y(min_id,1) = mutated_fitness;
       S(min_id,:) = mutated_x;
    else
       [~,max_id] = max(Y);
       Y(max_id,1) = mutated_fitness;
       S(max_id,:) = mutated_x;
    end
    % using the mutation to replace the worst individual in the population
    % or the worst personal best in the memory swarm
    [~,max_id] = max([PBest.Cost]);
    PBest(max_id).Position = mutated_x;
    PBest(max_id).Cost = mutated_fitness;
  end
```

Moreover, the "***Individual_Kriging***" used in "**KAGA**" for selecting the most promising offspring and updating the population and sample is given below

```
  function [S,Y,PBest,GlobalBest,mpc] = Individual_Kriging(S,Y,x,PBest,GlobalBest,mpc)
  %% Kriging individual based strategy
    global theta test_case test_function
    % calculate the weight for EI
    weight_ei = EDEIW(x,GlobalBest.Position);
    % estimate the fitness of swarm or population
    [y_hat, MSE] = kriging_predictor(S,Y,x,theta);
    % find the individual or particle of which has maximum weighted EI
    [~,max_EI_idx]= WMEI(MSE,y_hat,Y,weight_ei);
    % evalute the selected infill point
    infill_S = x(max_EI_idx,:);
    if test_case == 1
       [infill_Y,~,~] = numerical_Obj_Func(infill_S,test_function);
```

```
        mpc = [];
    elseif test_case == 2
        infill_Y = OPF_Obj_Func(infill_S,mpc);
    end
    % better-adoption and worse-abandonment" strategy for updating samples
    % and memory swarm (for pso) or population (for GA)
    if infill_Y <= GlobalBest.Cost
        GlobalBest.Cost = infill_Y;
        GlobalBest.Position = infill_S;
        [~,min_id] = min(Y);
        Y(min_id,1) = infill_Y;
        S(min_id,:) = infill_S;
    else
        [~,max_id] = max(Y);
        Y(max_id,1) = infill_Y;
        S(max_id,:) = infill_S;
    end
    % using the infill point to replace the worst individual in the population
    % or the worst personal best in the memory swarm
    [~,max_id] = max([PBest.Cost]);
    PBest(max_id).Position = infill_S;
    PBest(max_id).Cost = infill_Y;
end
```

Furthermore, the "***EDEIW***" function used in the "***Individual_Kriging***" script for calculating the Euclidean distance based weight is programmed below, while the "***kriging_predictor***" used to predict estimate the fitness of offspring and "***WMEI***" used to find the WMEI value and infill point are already given in A3 and A6, respectively.

```
function weight_ei = EDEIW(x,xg)
    %% Euclidean Distance based EI Weight
    % x := current swarm
    % xg := global best position
    d = sqrt(sum((x-xg).^2,2)); % euclidean distance
    du = max(d);
    dl = min(d);
    nd = (d-dl)./(du-dl); % normalized distance
    weight_ei = ( 1./(1+9.*exp(-4.4.*nd))); % between 0.1 and 0.9
end
```

Finally, the "***RPES***" used in the "***KAGA***" script for dealing with the reactive power constraints is given as below

```
function [final_fitness,lambda] = RPES(GlobalBest,mpc)
    %% Reactive Power Enforcement (RPE) strategy
    global PG BS VM VG TAP p1 p2 p3 p4 p5 p6 p7 p8 gen_pv vg_bus tap sh

    x = GlobalBest.Position;
    mpc.gen(gen_pv,PG)= x(p1:p2);
    mpc.bus(vg_bus,VM) = x(p3:p4);
    mpc.gen(:,VG) = x(p3:p4);
    mpc.branch(tap,TAP) = x(p5:p6);
    mpc.bus(sh,BS) = x(p7:p8);
    mpopt = mpoption;
    mpopt.pf.alg = 'NR';
    mpopt.pf.enforce_q_lims=1; % enable MATPOWER RPES
    [~,mpc.bus,mpc.gen,mpc.branch]=runpf(mpc,mpopt);
    x(p1:p2) = mpc.gen(gen_pv,PG);
    x(p3:p4) = mpc.bus(vg_bus,VM);
    x(p5:p6) = mpc.branch(tap,TAP);
    x(p7:p8) = mpc.bus(sh,BS);
    [final_fitness,~,lambda] = OPF_Obj_Func(x,mpc);

end
```

A9. Kriging Assisted Particle Swarm Optimization

The following Matlab codes of "***KAPSO***" can also be directly used to minimize the generation cost of the IEEE 118 bus test system, while properly changing "*test_case*", "*test_fucntion*", "*opf_system*" and "*opf_obj*" to solve different optimization problems.

```
function [final_fitness, lambda] = KAPSO
    addpath([cd '/matpower6.0/']);
    rng('default')
    %% test case preparation
    global nVar nPop VarMax VarMin theta sigma it MaxIt test_case test_function opf_system opf_obj
    % chose the test algorihtm
    test_alg = 1; % 0 means orginal PSO, 1 means Kriging assisted PSO
    % chose the test cases
    test_case = 2; % 1 means numerical function tests, 2 means OPF tests
    if test_case == 1
        % select the numberical functions, e.g. 'Ackley', 'Griewank', 'Levy', 'Rastrigin', 'Schwefel'
        test_function = 'Schwefel';
        [~,VarMin,VarMax] = numerical_Obj_Func([],test_function);
    elseif test_case == 2
        % select the test systems and the OPF objective functions
        opf_system = 'ieee_118_bus'; % or 'ieee_118_bus'
        opf_obj = 'cost'; % or 'losses'
        [VarMin,VarMax,mpc] = OPF_Initialization(opf_system);
    end
    % VarMin := Lower Bound of Variables, VarMax := Upper Bound of Variables
    %% PSO Parameters
    % rng('default')
    nVar = length(VarMin); % Number of control Variables
    VarSize = [1 nVar]; % Size of Decision Variables Matrix
    if test_case == 1
        MaxIt = 50000; % Maximum Number of Iterations
        nPop = 20; % Population Size (Swarm Size)
    elseif test_case == 2
        switch opf_system
            case 'ieee_30_bus'
                if test_alg == 0
                    MaxIt = 100; % Maximum Number of Iterations
                elseif test_alg == 1
                    MaxIt = 500; % Maximum Number of Iterations
                end
                nPop = 50; % Population Size (Swarm Size)
            case 'ieee_118_bus'
                if test_alg == 0
                    MaxIt = 1000; % Maximum Number of Iterations
                elseif test_alg == 1
                    MaxIt = 10000; % Maximum Number of Iterations
                end
                nPop = 100; % Population Size (Swarm Size)
        end
    end
    % Constriction Coefficients
    phi1 = 2.05;
    phi2 = 2.05;
    phi = phi1+phi2;
    chi = 2/(phi-2+sqrt(phi^2-4*phi));
    w = chi;          % Inertia Weight
    c1 = chi*phi1;    % Personal Learning Coefficient
    c2 = chi*phi2;    % Global Learning Coefficient
    % Velocity Limits
    VelMax = 0.1*(VarMax-VarMin);
    VelMin = -VelMax;
    %% Kriging and ELS parameters, tuning them can achieve better results
    if test_case == 1
```

```
        theta = 0.05;
        sigma = 1;
    elseif test_case == 2
        switch opf_system
            case 'ieee_30_bus'
                theta = 0.001;
            case 'ieee_118_bus'
                theta = 0.0017;
        end
        sigma = 0.01;
    end
    %% Initialization
    empty_particle.Position = [];
    empty_particle.Cost = [];
    empty_particle.Velocity = [];
    empty_particle_Best.Position = [];
    empty_particle_Best.Cost = [];

    particle = repmat(empty_particle,nPop,1); % current swarm
    PBest = repmat(empty_particle_Best,nPop,1); % memory swarm

    GlobalBest.Cost = inf; % global best particle

    % Kriging sample
    Y=zeros(nPop,1);
    S=zeros(nPop,nVar);

    for i = 1:nPop

        % Initialize Position
        particle(i).Position = unifrnd(VarMin,VarMax,VarSize);

        % Initialize Velocity
        particle(i).Velocity = zeros(VarSize);

        % Fitness Evaluation
        if test_case == 1 % numerical case
            [particle(i).Cost,~,~] = numerical_Obj_Func(particle(i).Position,test_function);
        elseif test_case == 2 % opf test
            [particle(i).Cost, mpc] = OPF_Obj_Func(particle(i).Position,mpc);
        end

        % Update Personal Best
        PBest(i).Position = particle(i).Position;
        PBest(i).Cost = particle(i).Cost;
        % Update Global Best
        if PBest(i).Cost < GlobalBest.Cost
            GlobalBest.Cost = PBest(i).Cost;
            GlobalBest.Position = PBest(i).Position;
        end

        % Update Samples
        S(i,:) = particle(i).Position;
        Y(i,1) = particle(i).Cost;

    end

    %% Main Optimization Loop
    for it = 1 : MaxIt
        % updating the particles
        for i = 1:nPop
            % Update Particle Velocity
            particle(i).Velocity = w*(particle(i).Velocity ...
                +c1*rand(VarSize).*(PBest(i).Position-particle(i).Position) ...
                +c2*rand(VarSize).*(GlobalBest.Position-particle(i).Position));

            % Apply Velocity Limits
```

```
            particle(i).Velocity = max(particle(i).Velocity,VelMin);
            particle(i).Velocity = min(particle(i).Velocity,VelMax);

            % Update Particle Position
            particle(i).Position = particle(i).Position + particle(i).Velocity;

            % Velocity Mirror Effect to fly back to the feasible region
            IsOutside = (particle(i).Position < VarMin | particle(i).Position > VarMax);
            particle(i).Velocity(IsOutside) = -particle(i).Velocity(IsOutside);

            % Apply Position Limits to enforce the control varibales within limits
            particle(i).Position = max(particle(i).Position,VarMin);
            particle(i).Position = min(particle(i).Position,VarMax);

            if test_alg == 0
               % Fitness Evaluation
               if test_case == 1 % numerical case
                  [particle(i).Cost,~,~] = numerical_Obj_Func(particle(i).Position,test_function);
               elseif test_case == 2 % opf test
                  [particle(i).Cost, mpc] = OPF_Obj_Func(particle(i).Position,mpc);
               end
               % Update Personal Best
               PBest(i).Position = particle(i).Position;
               PBest(i).Cost = particle(i).Cost;
               % Update Global Best
               if PBest(i).Cost < GlobalBest.Cost
                  GlobalBest.Cost = PBest(i).Cost;
                  GlobalBest.Position = PBest(i).Position;
               end
            end
         end
         %% perform the Individual based Kriging Strategy
         if  test_alg == 1
            x = reshape([particle.Position],nVar,nPop)'; % reshape the structure data "particle"
            if test_case == 1
               [S,Y,PBest,GlobalBest,~] = Individual_Kriging(S,Y,x,PBest,GlobalBest); % numerical test
            elseif test_case == 2
               [S,Y,PBest,GlobalBest,mpc] = Individual_Kriging(S,Y,x,PBest,GlobalBest,mpc); % opf test
            end
         end
         %% perform the Elitist Learning Strategy
         if test_case == 1 % numerical test
            [S,Y,PBest,GlobalBest,~] = ELS(S,Y,PBest,GlobalBest);
         elseif test_case == 2 % opf test
            [S,Y,PBest,GlobalBest,mpc] = ELS(S,Y,PBest,GlobalBest,mpc);
         end
         %% dispaly the current best found in the iteration "it"
         disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(GlobalBest.Cost)]);
      end
      %% ues Reactive Power Enforcement strategy to deal with the Qg violations
      if test_case == 1
         final_fitness = GlobalBest.Cost;
         lambda = [];
      elseif test_case == 2
         [final_fitness,lambda] = RPES(GlobalBest,mpc); % output the final value and penatly
      end
   end
```

Note that the "***numerical_Obj_Func***", "***OPF_Initialization***", "***OPF_Obj_Func***", "***ELS***", "***RPES***", "***Individual_Kriging***" and other related functions used in "***KAPSO***" script are already provided in A8.

A10. Pre-Selection Kriging Assisted Genetic Algorithm

Implementing the following Matlab m-file is to solve the OARPD problem of IEEE 57 bus test system by using “***PKAGA***” algorithm. By changing the input of term “*optimization_problme*” to “*OPF*” or “*Numerical*”, the script can be used to solve optimal power flow problem or numerical optimization problem, respectively. The input of "*algorithm_name*" can be changed to “*PKAGA*” or “*PKAPSO*” to select the desirable solution algorithm. The “***numerical_Obj_Func***” script used to calculate the objective value of different optimization functions is already given in A8. To solve the OPF problems, this script should be put in the same directory with OPF test bed [252] and the MATPOWER [244]. Changing the input of “*test_case*” to 0 or 1 to solve ORPD or OARPD problem, while changing the input of “*system*” to 41, 57, 118 or 300 to select the wanted OPF test system. Finally, changing the input of “*func*” to “Levy”, “Griewank”, “Rastrigin”, “Schwefel” or “Ackley” to select the needed numerical optimization function.

```
function test_numerical_and_OPF
restoredefaultpath
close all
clear all
clear global
clc
global proc ps mpc res fhd func test_case scenario trial args ns nPop ncPop nVar VarMin VarMax
optimization_problme
algorithm_name = 'PKAGA'; % or 'aka_pso'
algorithm_hd = str2func(algorithm_name);
%% Select the optimization problme: either Numerical Optimization or OPF
optimization_problme = 'OPF'; % 'OPF' or 'Numerical'
switch optimization_problme
    case 'OPF'
        %% OPF Test Bed
        % Task Force on Modern Heuristic Optimization Test Beds
        % Working Group on Modern Heuristic Optimization
        % Intelligent Systems Subcommittee
        % Power System Analysis, Computing, and Economic Committee
        %
        % Sebastian Wildenhues (E-Mail: sebastian.wildenhues@uni-due.de)
        % 18th September 2013
        %
        % Application of Modern Heuristic Optimization Algorithms
        % for Solving Optimal Power Flow Problems
        %
        % Test bed declarations V1.0
        %
        % Employing MATPOWER as underlying power flow and basic Particle Swarm
        % Evaluation (PSO) algorithm as optimization engine, you can use the
        % test bed declarations as the template shown below.
        %
        % Results will be buffered and agglomerated automatically for storage to
        % formatted ASCII-files. Refer to problem definitions and implementation
        % guidelines for details.
        %
        % Note:
        % This implementation has been tested using various MATLAB versions and
        % hardware platforms. Feel free to contact us in case of incompatibilities.
        % A MATPOWER installation must be on the MATLAB search path.
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
        % INITIALIZE simulation parameters
        % Possible values for test_case:
```

```
% 0 Optimal Reactive Power Dispatch
% 1 Optimal Active and Reactive Power Dispatch
test_case = 1; % define the OPF test objective function
% Possible values for system:
% 41 (WPP), 57, 118 and 300
system = 57; % define the test system
switch system
    case 41
        nPop = 20; % population/swarm size
        ncPop = 10; % number of candidates for pre-selection
        ns = 20; % Kriging sample size
    case 57
        nPop = 30;
        ncPop = 20;
        ns = 30;
    case 118
        nPop = 100;
        ncPop = 30;
        ns = 100;
    case 300
        nPop = 150;
        ncPop = 50;
        ns = 150;
    otherwise
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% input arguments
run_in_parallel = 0;
show_lf_info = 0;
refresh = 1000;
args{1} = system;
args{2} = show_lf_info;
args{3} = 1;
args{4} = refresh;
args{5} = algorithm_name;
args{6} = run_in_parallel;
args{7} = [];
args{8} = [];
v = ver;

stop_test_case = 0;
while ~stop_test_case
    test_case = test_case + 1;
    scenario = 0;
    stop_scenario = 0;
    while ~stop_scenario
        % Possible values for scenario:
        % 1:    57, 118 and 300 bus systems
        % 1-96: 41 bus (WPP) system
        scenario = scenario + 1;
        [stop_test_case,stop_scenario,err,obs] = test_bed_OPF(test_case,scenario,1,args);
        args{7} = stop_test_case; % Flag indicating the last test case for current system.
        args{8} = stop_scenario; % Flag indicating the last test case for current system.
        proc.consider_contingencies = 0;
        proc.contingencies = [];
        nVar = ps.D;
        VarMin = ps.x_min;
        VarMax = ps.x_max;
        if ~err
            for trial = 1 : proc.n_run
                test_bed_OPF(test_case,scenario,trial,args);
                feval(algorithm_hd);
                fprintf('Run %d finished.\n',trial);
            end
            test_bed_OPF(test_case,scenario,987,args);
        end
    end
```

```
        end
    case 'Numerical'
        %% Numerical Optimization tests
        % chose the Numerical optimization function 'Levy' 'Griewank' 'Rastrigin' 'Schwefel' 'Ackley'
        func = 'Ackley';
        % obtain the lower and upper bounds of variables
        [~,VarMin,VarMax] = numerical_Obj_Func([],func);
        nPop = 30; % population/swarm size
        ncPop = 30; % number of candidates for pre-selection
        ns = 30; % Kriging sample size
        nVar = 50; % number of variables
        GlobalBest = feval(algorithm_hd);
end
end
```

The script of "***PKAGA***" is given as below

```
function GlobalBest = PKAGA()
%% Pre-selection Kriging Assisted Genetic Algorithm
%% Problem Definition
global optimization_problme func
global proc mpc test_case scenario trial args
global FEs MaxFEs OA row AW_OA AW_WEI nPop ns S Y nVar VarMin VarMax theta VarRange
% rng('default')
switch optimization_problme
    case 'OPF'
        proc.i_eval = 0;
        % Function evaluation at which
        % the last update of the global
        % best solution occurred.
        % Refers to the internal evaluation
        % using static penalty constraint
        % handling method.
        proc.last_improvement = 1;
        % Signalizing your
        % to stop running.
        proc.finish = 0;
        FEs = proc.i_eval;
        MaxFEs = proc.n_eval;
    case 'Numerical'
        FEs = 0;
        MaxFEs = 300000;
end
VarRange = VarMax - VarMin;
VarSize = [1 nVar];   % Decision Variables Matrix Size

%% GA Parameters
pc = 0.7;                % Crossover Percentage
nc = 2*round(pc*nPop/2);  % Number of Offspring (also Parents)
pm = 0.3;                % Mutation Percentage
nm = round(pm*nPop);      % Number of Mutants
%% Initialization
% sampel set
S = zeros(ns,nVar);
Y = zeros(ns,1);
% population set
empty_individual.Position = [];
empty_individual.Cost = [];
pop = repmat(empty_individual,nPop,1);
switch optimization_problme
    case 'OPF'
        if proc.system == 300 && test_case == 2
            pd = sum(mpc{1,1}.bus(:,3)) + 450;
            ref_max = mpc{1,1}.gen(56,9); % slack bus
        end
end
for i = 1:nPop
```

```
    % Initialize Position
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
    % it was found that the sum of uniformly random generated active
    % outputs may far less than the total load demands for IEEE 300 bus
    % system. This would easily result in convergence issue of power flow analysis,
    % therefore the following strategy is used to ensure that the randomly generated total
    % generations larger than total loads
    switch optimization_problme
        case 'OPF'
            if proc.system == 300 && test_case == 2
                if (sum(pop(i).Position(1:68))+ref_max) < pd
                    pop(i).Position(1:68) = pop(i).Position(1:68)...
                        + 0.5.*(VarMax(1:68)-pop(i).Position(1:68)).*rand(1,68);
                end
                pop(i).Position = max(pop(i).Position,VarMin);
                pop(i).Position = min(pop(i).Position,VarMax);
            end
    end
    % Evaluation
    switch optimization_problme
        case 'OPF'
            [pop(i).Cost,~,~,pop(i).Position] = test_bed_OPF(test_case,scenario,trial,args,pop(i).Position);
        case 'Numerical'
            pop(i).Cost = numerical_Obj_Func(pop(i).Position,func);
    end
    FEs = FEs + 1;
    % update sample
    S(i,:) = pop(i).Position;
    Y(i,1) = pop(i).Cost;
end
% Sort Population
Costs = [pop.Cost];
[~, SortOrder] = sort(Costs);
pop = pop(SortOrder);
GlobalBest = pop(1);
% Initialization of Award of each crossover and mutation
Cross_Award = zeros(nc/2,1);
Muta_Award = zeros(nm,1);
% initialize the Orthogonal Array
[OA,row] = orth_array(2,nVar);
% kriging theta parameter
theta = 1;
% Threshold Award Values for triggering different operators
% and assigning the weight to Expected Improvement.
switch optimization_problme
    case 'OPF'
        if proc.system == 41
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 5;
        elseif proc.system == 57
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 10;
        elseif proc.system == 118
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 20;
        elseif proc.system == 300
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 30;
        end
    case 'Numerical'
        AW_WEI = 0;
        AW_KELS = 0;
        AW_OA = 10;
```

```
end
while FEs < MaxFEs
    %% Crossover
    popc = repmat(empty_individual,nc/2,2);
    for k=1:nc/2
        % Select Parents
        gb = GlobalBest.Cost; % current best is always selected as parent 1
        i2 = Selection(Costs);
        if isempty(i2)
            i2 = randi(nPop);
        end
        p1 = GlobalBest;
        p2 = pop(i2);
        % Apply Crossover
        [popc(k,1).Position, popc(k,2).Position,Cross_Award(k)] =
Kriging_Pre_Selection_Crossover(Cross_Award(k),p1.Position,p2.Position,GlobalBest);
        % Evaluate best candidate Offspring 1
        switch optimization_problme
            case 'OPF'
                [popc(k,1).Cost,~,~,popc(k,1).Position] = test_bed_OPF(test_case...
                    ,scenario,trial,args,popc(k,1).Position);
            case 'Numerical'
                popc(k,1).Cost = numerical_Obj_Func(popc(k,1).Position,func);
        end
        FEs = FEs + 1;
        % Evaluate best candidate Offspring 2
        switch optimization_problme
            case 'OPF'
                [popc(k,2).Cost,~,~,popc(k,2).Position] = test_bed_OPF(test_case...
                    ,scenario,trial,args,popc(k,2).Position);
            case 'Numerical'
                popc(k,2).Cost = numerical_Obj_Func(popc(k,2).Position,func);
        end
        FEs = FEs + 1;
        % find best offspring
        [~,id] = min([popc(k,1).Cost;popc(k,2).Cost]);
        best_offspring = popc(k,id);
        % iteratively rolling replacement strategy for updating samples
        Y(nPop+1:nPop+2) = [popc(k,1).Cost;popc(k,2).Cost];
        S(nPop+1:nPop+2,:) = [popc(k,1).Position;popc(k,2).Position];
        S(1:2,:) = [];
        Y(1:2,:) = [];
        % update global best and crossover award information
        if best_offspring.Cost < GlobalBest.Cost
            Cross_Award(k) = 0;
            GlobalBest.Cost = best_offspring.Cost;
            GlobalBest.Position = best_offspring.Position;
        else
            Cross_Award(k) = Cross_Award(k)+1;
        end
        %% elitism learning
        if Cross_Award(k) > AW_KELS
            [GlobalBest,Cross_Award(k)] = KELS(Cross_Award(k),GlobalBest);
        end
        %% OA
        if best_offspring.Cost < p1.Cost || best_offspring.Cost < p2.Cost...
                || GlobalBest.Cost < gb && best_offspring.Cost ~= GlobalBest.Cost
            [popc(k,id),GlobalBest,Cross_Award(k)] = Kriging_OA(Cross_Award(k),popc(k,id),GlobalBest);
        end
    end
    popc = popc(:);
    %% Mutation
    popm = repmat(empty_individual,nm,1);
    for k = 1:nm
        gb = GlobalBest.Cost;
        % Select offspring to perform mutation
        i = Selection([popc.Cost]);
```

```
        if isempty(i)
            i = randi(nc);
        end
        p = popc(i);
        % Apply Mutation
        popm(k).Position = Kriging_Pre_Selection_Mutation(p.Position,GlobalBest);
        % Evaluate Mutant
        switch optimization_problme
            case 'OPF'
                [popm(k).Cost,~,~,popm(k).Position] = test_bed_OPF(test_case...
                    ,scenario,trial,args,popm(k).Position);
            case 'Numerical'
                popm(k).Cost = numerical_Obj_Func(popm(k).Position,func);
        end
        FEs = FEs + 1;
        % update global best and award information
        if popm(k).Cost < GlobalBest.Cost
            Muta_Award(k) = 0;
            GlobalBest.Cost = popm(k).Cost;
            GlobalBest.Position = popm(k).Position;
        else
            Muta_Award(k) = Muta_Award(k) + 1;
        end
        % iteratively rolling replacement strategy for updating samples
        Y(nPop+1) = popm(k).Cost;
        S(nPop+1,:) = popm(k).Position;
        S(1,:) = [];
        Y(1,:) = [];
        %% elitism learning
        if Muta_Award(k) > AW_KELS
            [GlobalBest,Muta_Award(k)] = KELS(Muta_Award(k),GlobalBest);
        end
        %% OA
        if popm(k).Cost < p.Cost || GlobalBest.Cost < gb && popm(k).Cost ~= GlobalBest.Cost
            [popm(k),GlobalBest,Muta_Award(k)] = Kriging_OA(Muta_Award(k),popm(k),GlobalBest);
        end
    end
    % Create Merged Population
    pop = [pop
        popc
        popm];
    % Sort Population
    Costs=[pop.Cost];
    [Costs, SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Truncation
    Costs = Costs(1:nPop);
    pop = pop(1:nPop);
    % check whether the global best is included in the new population
    if GlobalBest.Cost < pop(1).Cost
        pop(1) = GlobalBest;
    end
    switch optimization_problme
        case 'Numerical'
            disp(['Current Number of Function Evaluations ' num2str(FEs)...
                ': Best Cost = ' num2str(GlobalBest.Cost)]);
    end
    switch optimization_problme
        case 'OPF'
            if proc.finish
                return;
            end
    end
end
end
```

The script modified to deal with the constraints of OPF is given as below

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task Force on Modern Heuristic Optimization Test Beds
% Working Group on Modern Heuristic Optimization
% Intelligent Systems Subcommittee
% Power System Analysis, Computing, and Economic Committee
%
% Sebastian Wildenhues (E-Mail: sebastian.wildenhues@uni-due.de)
% 14th February 2014
%
% Application of Modern Heuristic Optimization Algorithms
% for Solving Optimal Power Flow Problems
%
% Incorporating static penalty constraint handling method.
%
% This routine is called subsequent to every function evaluation,
% i.e. power flow calculation. It does not affect the calculations
% done in test_bed_OPF.p, which calculates internally the fitness by
% using static penalty constraint handling method.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [f,g]=constraint_handling(o,g)
  % You may use procedural information
  % such as function evaluation counter
  % proc.i_eval to dynamically adjust
  % your constraint handling method.
  global proc ps mpc constraints
  constraints = g;
  switch proc.system
    case 41
      k = [1e3 1 1];
      n_vol = 2*ps.n_bus;
      n_br = numel(mpc{1}.branch(:,1));

      g_vol = sum(g(1:n_vol));
      g_br = sum(g(1+n_vol:n_vol+n_br));
      g_slark = g(end);

      penalty = k(1)*g_vol + k(2)*g_br + k(3)*g_slark;

    case 57
      if proc.test_case == 1
        k = [1e3 0.01 0.01];
        n_vol = 2*ps.n_load;
        n_br = numel(mpc{1}.branch(:,1));

        g_vol = sum(g(1:n_vol));
        g_br = sum(g(1+n_vol:n_vol+n_br));
        g_q_gen = sum(g(1+n_vol+n_br:end));

        penalty = k(1)*g_vol + k(2)*g_br + k(3)*g_q_gen;
      else
        k = [1 1e3 1 1];
        n_vol = 2*ps.n_load;
        n_br = numel(mpc{1}.branch(:,1));

        g_slark = g(1);
        g_vol = sum(g(2:n_vol+1));
        g_br = sum(g(2+n_vol:n_vol+n_br+1));
        g_q_gen = sum(g(2+n_vol+n_br:end));

        penalty = k(1)*g_slark + k(2)*g_vol + k(3)*g_br + k(4)*g_q_gen;
      end
    case 118
      if proc.test_case == 1
```

```
            k = [1e3 0.1 0.1];
            n_vol = 2*ps.n_load;
            n_br = numel(mpc{1}.branch(:,1));

            g_vol = sum(g(1:n_vol));
            g_br = sum(g(1+n_vol:n_vol+n_br));
            g_q_gen = sum(g(1+n_vol+n_br:end));

            penalty = k(1)*g_vol + k(2)*g_br + k(3)*g_q_gen;
        else
            k = [1 1e3 1 1];
            n_vol = 2*ps.n_load;
            n_br = numel(mpc{1}.branch(:,1));

            g_slark = g(1);
            g_vol = sum(g(2:n_vol+1));
            g_br = sum(g(2+n_vol:n_vol+n_br+1));
            g_q_gen = sum(g(2+n_vol+n_br:end));

            penalty = k(1)*g_slark + k(2)*g_vol + k(3)*g_br + k(4)*g_q_gen;
        end
    case 300
        if proc.test_case == 1
            k = [1e3 1 1];
            n_vol = 2*ps.n_load;
            n_br = numel(mpc{1}.branch(:,1));

            g_vol = (sum(g(1:n_vol)));
            g_br = sum(g(1+n_vol:n_vol+n_br));
            g_q_gen = sum(g(1+n_vol+n_br:end));

            penalty = k(1)*g_vol + k(2)*g_br + k(3)*g_q_gen;
        else
            k = [1 1e3 1 1];
            n_vol = 2*ps.n_load;
            n_br = numel(mpc{1}.branch(:,1));

            g_slark = g(1);
            g_vol = sum(g(2:n_vol+1));
            g_br = sum(g(2+n_vol:n_vol+n_br+1));
            g_q_gen = sum(g(2+n_vol+n_br:end));

            penalty = k(1)*g_slark + k(2)*g_vol + k(3)*g_br + k(4)*g_q_gen;
        end
    otherwise
        % This must never happen!
end

% Uniform penalty coefficient.
% Sum up all violations.
% Note that, at this point, constraint
% vector g consists only of elements
% >=0 due to g(g<1e-4)=0 as internal
% preliminary step.
%     g = 1e7*sum(g.^2);
g = 1;
% Fitness function.

f = o + penalty;
end
```

The "***Kriging_Pe-Selection_Crossover***" is programmed as

```
function [y1,y2,Award] = Kriging_Pre_Selection_Crossover(Award,x1,x2,GlobalBest)
%% Kriging Pre-selection Crossover
global ncPop theta S Y AW_WEI nVar VarMin VarMax
%% single Point Crossover
```

```
    if ncPop >= nVar - 1
        ncPop = nVar - 1;
        % randomly generate crossover point for each candidate offspring
        cp = 1:ncPop;
    else
        cp = randsample(nVar - 1,ncPop);
    end
    % offspring 1 candidates
    y1 = zeros(ncPop,nVar);
    for s = 1 : ncPop
        y1(s,:) = [x1(1:cp(s)) x2(cp(s)+1:end)];
    end
    % offspring 2 candidates
    y2 = zeros(ncPop,nVar);
    for s = 1 : ncPop
        y2(s,:) = [x2(1:cp(s)) x1(cp(s)+1:end)];
    end
%% Arithmetic Crossover
%     gamma = 0.5;
%     alpha = unifrnd(-gamma,1+gamma,ncPop,nVar);
%     % offspring 1 candidates
%     y1=alpha.*x1+(1-alpha).*x2;
%         index = false(ncPop,1);
%     % remove the infesiable offspring 1 candidates
%     for j = 1 : ncPop
%         if all(y1(j,:) >= VarMin & y1(j,:) <= VarMax)
%             index(j) = true;
%         else
%             index(j) = false;
%         end
%     end
%     if any(index)
%         y1 = y1(index,:);
%     % if all candidates are infeasible, random select one candidate and enforce
%     % its limits as offspring 1
%     else
%         id = randi(ncPop);
%         y1 = y1(id,:);
%         y1 = max(y1,VarMin);
%         y1 = min(y1,VarMax);
%     end
%     % offspring 2 candidates
%     y2=alpha.*x2+(1-alpha).*x1;
%     for j = 1 : ncPop
%         if all(y2(j,:) >= VarMin & y2(j,:) <= VarMax)
%             index(j) = true;
%         else
%             index(j) = false;
%         end
%     end
%     % remove the infesiable offspring 2 candidates
%     if any(index)
%         y2 = y2(index,:);
%     % if all candidates are infeasible, random select one candidate and enforce
%     % its limits as offspring 2
%     else
%         id = randi(ncPop);
%         y2 = y2(id,:);
%         y2 = max(y2,VarMin);
%         y2 = min(y2,VarMax);
%     end
%% Kriging Pre-selection
    % kriging select best candidate offspring 1
    % if only one candidate, thus no need Kriging pre-selection
    if size(y1,1) > 1
        [YX,MSE] = kriging_predictor(S,Y,y1,theta);
        if Award > AW_WEI
```

```
            weight_ei = 0;
        else
            weight_ei = 1;
        end
        [~,id] = WMEI(MSE,YX,GlobalBest.Cost,weight_ei);
        y1 = y1(id,:);
    end
    % kriging select best candidate offspring 2
    % if only one candidate, thus no need Kriging pre-selection
    if size(y2,1) > 1
        [YX,MSE] = kriging_predictor(S,Y,y2,theta);
        if Award > AW_WEI
            weight_ei = 0;
        else
            weight_ei = 1;
        end
        [~,id] = WMEI(MSE,YX,GlobalBest.Cost,weight_ei);
        y2 = y2(id,:);
    end
end
```

The Kriging pre-selection mutation is given below

```
function y = Kriging_Pre_Selection_Mutation(x,GlobalBest)
%% Kriging Pre-selection Crossover
    global FEs MaxFEs ps ncPop theta S Y nVar VarMax VarMin optimization_problme
    d = randi(nVar);
    y = x;
%% continuous variable
    switch optimization_problme
        case 'Numerical'
            ps.x_type(d) = 0;
    end
    if ps.x_type(d) == 0
        mutation_positions = zeros(1,ncPop);
        % Non-Uniform Mutation, nonlinear decrease along with function evaluations
        b = 1; % the larger the b the larger the (1-rand^((1-It/MaxIt)^b)) tend to be
        for s = 1 : ncPop
            if rand > 0.5
                mutation_positions(s) = x(d) + 0.1*(VarMax(d) - VarMin(d))...
                    *(1 - rand^(1 - (FEs/MaxFEs))^b);
            else
                mutation_positions(s) = x(d) - 0.1*(VarMax(d) - VarMin(d))...
                    *(1 - rand^(1 - (FEs/MaxFEs))^b);
            end
        end
        % Gaussian Mutation
%       for s = 1 : ncPop
%           mutation_positions(s) = x(d) + 0.1 * (VarMax(d) - VarMin(d)) * randn;
%       end
        idx = find(mutation_positions <= VarMax(d) & mutation_positions >= VarMin(d));
        if ~isempty(idx)
            % remove the infeasible mutation candidates
            mutation_particles = repmat(x,length(idx),1);
            mutation_particles(:,d) = mutation_positions(idx);
            % Kriging pre-selection
            [YX,MSE] = kriging_predictor(S,Y,mutation_particles,theta);
            [~,id] = WMEI(MSE,YX,GlobalBest.Cost,1);
            y = mutation_particles(id,:);
        else
            % no feasible mutation, no need Kriging pre-selection
            % thus random select on candidate and enforce the limits
            id = randi(ncPop);
            y(d) = mutation_positions(id);
            y(d) = max(y(d),VarMin(d));
            y(d) = min(y(d),VarMax(d));
        end
```

```
%% discrete variable
    elseif ps.x_type(d) == 1
        if round(x(d)) == VarMax(d)
            y(d) = x(d) - 1;
        elseif round(x(d)) == VarMin(d)
            y(d) = x(d) + 1;
        else
            if rand > 0.5
                y(d) = x(d) + 1;
            else
                y(d) = x(d) - 1;
            end
        end
        y(d) = max(y(d),VarMin(d));
        y(d) = min(y(d),VarMax(d));
%% binary variable
    elseif ps.x_type(d) == 2
        if round(x(d)) == 1
            y(d) = 0;
        else
            y(d) = 1;
        end
    end
end
```

The "***selection***" operator used in "***PKAGA***" has already been given in the appendix A8. while the "*kriging_predictor*" used to predict estimate the fitness of offspring and "***WMEI***" used to find the WMEI value and infill point are already given in A3 and A6, respectively. Note, the sample "*S*" and predicting point "*x*" used in the "***kriging_predictor***" is no longer normalized to mean and standard deviation values when solving OPF problems. Instead, "*S*" and "*x*" normalize to $S = \frac{S-VarMin}{VarMax-VarMin}$ $x = \frac{x-VarMin}{VarMax-VarMin}$, respectively, to achieve better performance, where $VarMin$ and $VarMax$ are the lower and upper bounds of the variable. The "***KELS***" operator is coded as

```
function [GlobalBest,Award] = KELS(Award,GlobalBest)
%% Kriging Pre-selection Elitism Learning Strategy
    global FEs MaxFEs func ps theta S Y ns ncPop test_case scenario trial args nVar VarMin VarMax
optimization_problme
    switch optimization_problme
        case 'OPF'
            sigma = 0.1;
        case 'Numerical'
            sigma = 1 - 0.9*FEs/MaxFEs;
    end
    % random select a varibale to perform ELS mutation
    d = randi(nVar);
%% generate multiple mutation candidates
    % for continuous variable
    switch optimization_problme
        case 'Numerical'
            ps.x_type(d) = 0;
    end
    if ps.x_type(d) == 0
        mutation_positions = GlobalBest.Position(d) + normrnd(0,sigma,1,ncPop).*(VarMax(d) - VarMin(d));
    % for discrete variable, no need pre-selection
    elseif ps.x_type(d) == 1
        if round(GlobalBest.Position(d)) == VarMax(d)
            mutation_positions = GlobalBest.Position(d) - 1;
        elseif round(GlobalBest.Position(d)) == VarMin(d)
            mutation_positions = GlobalBest.Position(d) + 1;
        else
            if rand > 0.5
                mutation_positions= GlobalBest.Position(d) + 1;
```

```
            else
                mutation_positions = GlobalBest.Position(d) - 1;
            end
        end
        % enforce limits
        mutation_positions = max(mutation_positions,VarMin(d));
        mutation_positions = min(mutation_positions,VarMax(d));
    % for binary variable, no need pre-selection
    elseif ps.x_type(d) == 2
        if round(GlobalBest.Position(d)) == 1
            mutation_positions = 0;
        else
            mutation_positions = 1;
        end
    end
%% select the best candidate
    % find the feasible mutation values to save computational cost
    % if all mutations are infeasible, skip the Kriging pre-selection and
    % function evaluation
    idx = find(mutation_positions <= VarMax(d) & mutation_positions >= VarMin(d));
    if ~isempty(idx)
        n_idx = length(idx);
        mutation_particles = repmat(GlobalBest.Position,n_idx,1);
        mutation_particles(:,d) = mutation_positions(idx);
        if n_idx == 1 % no need Kriging Pre-selection
            id = 1;
        else  % perform Kriging Pre-selection
            [YX,MSE] = kriging_predictor(S,Y,mutation_particles,theta);
            [~,id]= WMEI(MSE,YX,GlobalBest.Cost,1);
        end
        % evaluate mutation
        KELS_position = mutation_particles(id,:);
        switch optimization_problme
            case 'OPF'
                [KELS_cost,~,~,KELS_position] = test_bed_OPF(test_case,scenario,trial,args,KELS_position);
            case 'Numerical'
                KELS_cost = numerical_Obj_Func(KELS_position,func);
        end
        FEs = FEs + 1;
        % iteratively rolling replacement strategy for updating samples
        Y(ns+1) = KELS_cost;
        S(ns+1,:) = KELS_position;
        S(1,:) = [];
        Y(1,:) = [];
        % update award and global best information
        if KELS_cost < GlobalBest.Cost
            Award = 0;
            GlobalBest.Cost = KELS_cost;
            GlobalBest.Position = KELS_position;
        end
    end
end
```

The "***Kriging_OA***" used to perform KOEDLS operator is scripted as

```
function [Selected,GlobalBest,Award] = Kriging_OA(Award,Selected,GlobalBest)
%% OA combinations between two selected candidates
global FEs theta OA row AW_OA ns S Y test_case scenario trial args nVar optimization_problme func
% candidate 1, selected particle personal best (PSO), or selected
% offspring (GA)
Xp = Selected.Position; % candidate 1, selected particle personal best
Xg = GlobalBest.Position; % candidate 2, global best
OA_candidates_position = zeros(row,nVar);
Xg_1 = repmat(Xg,row,1);
Xp_2 = repmat(Xp,row,1);
level_1_index = OA==0;
level_2_index = OA==1;
```

```
% all possiable OA combinations between "Xp" and "Xg"
OA_candidates_position(level_1_index) = Xg_1(level_1_index);
OA_candidates_position(level_2_index) = Xp_2(level_2_index);
%% evaluate or estimate the OA candidates
if Award > AW_OA
    % use power flow to evaluate all combinations
    OA_candidates_cost = zeros(row,1);
    for j = 1 : row
        % calculation the fitness of OA combinations and avoid unnecessary FEs
        if all(OA_candidates_position(j,:) == Xg)
            OA_candidates_cost(j,1) = GlobalBest.Cost;
        elseif all(OA_candidates_position(j,:) == Xp)
            OA_candidates_cost(j,1) = Selected.Cost;
        else
            switch optimization_problme
                case 'OPF'
                    [OA_candidates_cost(j,1),~,~,OA_candidates_position(j,:)] = ...
                        test_bed_OPF(test_case,scenario,trial,args,OA_candidates_position(j,:));
                case 'Numerical'
                    OA_candidates_cost(j,1) = numerical_Obj_Func(OA_candidates_position(j,:),func);
            end
            FEs = FEs + 1;
        end
    end
else
    % use Kriging to estimate all combinations
    [YX,~] = kriging_predictor(S,Y,OA_candidates_position,theta);
    % but this work used estimations
    OA_candidates_cost = YX; % candidates estimations
end
%% Factor Analysis
% Factor Analysis based on Kriging estimation values
% one could try to use EI values for Factor Analysis
effect_factor_1 = (level_1_index'*OA_candidates_cost)'./sum(level_1_index);
effect_factor_2 = (level_2_index'*OA_candidates_cost)'./sum(level_2_index);
effect_factor = [effect_factor_1;effect_factor_2];
% find the minimum effect factors for each variable
[~,index] = min(effect_factor);
% define the best combination from OA candidates
index_array_1 = (index==1);
index_array_2 = (index==2);
best_OA_particle(index_array_1) = Xg(index_array_1);
best_OA_particle(index_array_2) = Xp(index_array_2);
% calculation the fitness of best OA combinations and avoid unnecessary FEs
if Award > AW_OA
    [logic,idx] = ismember(best_OA_particle,OA_candidates_position,'rows');
    if logic
        best_OA_cost = OA_candidates_cost(idx,1);
    else
        switch optimization_problme
            case 'OPF'
                [best_OA_cost,~,~,best_OA_particle] = test_bed_OPF...
                    (test_case,scenario,trial,args,best_OA_particle);
            case 'Numerical'
                best_OA_cost = numerical_Obj_Func(best_OA_particle,func);
        end
        FEs = FEs + 1;
    end
    % calculation the fitness of best OA combinations
    % if the best combination is same as Xp or Xg
    % the fitness of best combination is directly assigned to the
    % corresponding value to save computational cost
else
    if all(best_OA_particle == Xp)
        best_OA_cost = Selected.Cost;
    elseif all(best_OA_particle == Xg)
        best_OA_cost = GlobalBest.Cost;
```

```
    else % otherwise use power flow analysis to evaluate the best combination
        switch optimization_problme
            case 'OPF'
                [best_OA_cost,~,~,best_OA_particle] = test_bed_OPF...
                    (test_case,scenario,trial,args,best_OA_particle);
            case 'Numerical'
                best_OA_cost = numerical_Obj_Func(best_OA_particle,func);
        end
        FEs = FEs + 1;
    end
end
%% iteratively rolling replacement strategy for updating samples
Y(ns+1) = best_OA_cost;
S(ns+1,:) = best_OA_particle;
S(1,:) = [];
Y(1,:) = [];
%% Update selected Personal Best(PSO),or selected offspring(GA),Global Best and award
if best_OA_cost < Selected.Cost
    Selected.Cost = best_OA_cost;
    Selected.Position = best_OA_particle;
    if best_OA_cost < GlobalBest.Cost
        Award = 0;
        GlobalBest.Cost = best_OA_cost;
        GlobalBest.Position = best_OA_particle;
    end
end
end
```

Finally, the "***orth_array***" used in "***PKAGA***" for providing orthogonal array can be found below

```
function [OA,row] = orth_array(Q,nVar)
m = floor(log(nVar*Q-nVar+1)/log(Q));
if (Q^m-1)/(Q-1) < nVar
    m = ceil(log(nVar*Q-nVar+1)/log(Q));
end

row = Q^m; % or row=Q^ceil(log(nVar+1)/log(Q));
col = (Q^m-1)/(Q-1);

OA = zeros(row,col);

%% construct OAs
% Compute the basic columns
for k = 1:m
    j=((Q^(k-1)-1)/(Q-1))+1;
    for i = 1:Q^m
        OA(i,j)=floor(((i-1)/(Q^(m-k)))); % I have to use floor to get the correct result
    end
end

% Compute the non basic columns
for k = 2:m
    j=((Q^(k-1)-1)/(Q-1))+1;
    for s = 1:j-1
        for t = 1:Q-1
            x=j+(s-1)*(Q-1)+t;
            OA(:,x)=mod(OA(:,s)*t+OA(:,j),Q);
        end
    end
end
OA=mod(OA,Q);

% in order to remove the limitation, the following arguments are added:
if nVar ~= (Q^m-1)/(Q-1)
    OA = OA(:,1:nVar);
end
end
```

A11. Pre-selection Kriging Assisted Particle Swarm Optimization

The "***PKAPSO***" script is given below

```
function GlobalBest = PKAPSO()
%% Pre-selection Kriging Assisted Particle Swarm Optimization
%% Problem Definition
global optimization_problme func
global proc ps mpc test_case scenario trial args
global FEs MaxFEs OA row Award AW_OA nPop ns ncPop S Y nVar VarMin VarMax theta VarRange
% rng('default')
switch optimization_problme
    case 'OPF'
        proc.i_eval = 0;
        % Function evaluation at which
        % the last update of the global
        % best solution occurred.
        % Refers to the internal evaluation using static penalty constraint handling method.
        proc.last_improvement = 1;
        % Signalizing your to stop running.
        proc.finish = 0;
        FEs = proc.i_eval;
        MaxFEs = proc.n_eval;
    case 'Numerical'
        FEs = 0;
        MaxFEs = 300000;
end
VarRange = VarMax - VarMin;
VarSize = [1 nVar];   % Decision Variables Matrix Size
%% PSO Parameters
% Constriction Coefficients
phi1 = 2.05;
phi2 = 2.05;
phi = phi1+phi2;
chi = 2/(phi-2+sqrt(phi^2-4*phi));
w = chi;          % Inertia Weight
c2 = chi*phi2;    % Global Learning Coefficient

% Velocity Limits
switch optimization_problme
    case 'OPF'
        VelMax = 0.5.*(VarMax-VarMin);
        VelMin = -VelMax;
        VelMax(ps.x_type==2) = VelMax(ps.x_type==2).*2;
        VelMin(ps.x_type==2) = VelMin(ps.x_type==2).*2;
    case 'Numerical'
        VelMax = 0.5.*(VarMax-VarMin);
        VelMin = -VelMax;
end
%% Initialization
empty_particle.Position = [];
empty_particle.Cost = [];
empty_particle.Velocity = [];
Particle = repmat(empty_particle,nPop,1);
empty_particle_Best.Position = [];
empty_particle_Best.Cost = [];
PBest = repmat(empty_particle_Best,nPop,1);
GlobalBest.Cost = inf;
switch optimization_problme
    case 'OPF'
        if proc.system == 300 && test_case == 2
            pd = sum(mpc{1,1}.bus(:,3)) + 450;
            ref_max = mpc{1,1}.gen(56,9); % slack bus
        end
end
% sample set
```

```
S = zeros(ns,nVar);
Y = zeros(ns,1);
for i=1:nPop
    % Initialize Position
    Particle(i).Position = unifrnd(VarMin,VarMax,VarSize);
    % it was found that the sum of uniformly random generated active
    % outputs may far less than the total load demands for IEEE 300 bus
    % system. This would easily result in convergence issue of power flow analysis,
    % therefore the following strategy is used to ensure that the randomly generated total
    % generations larger than total loads
    switch optimization_problme
        case 'OPF'
            if proc.system == 300 && test_case == 2
                if (sum(Particle(i).Position(1:68))+ref_max) < pd
                    Particle(i).Position(1:68) = Particle(i).Position(1:68)...
                        + 0.5.*(VarMax(1:68)-Particle(i).Position(1:68)).*rand(1,68);
                end
                % enforce the limits
                Particle(i).Position(1:68) = max(Particle(i).Position(1:68),VarMin(1:68));
                Particle(i).Position(1:68) = min(Particle(i).Position(1:68),VarMax(1:68));
            end
    end
    % Initialize Velocity
    Particle(i).Velocity = zeros(VarSize);
    % Evaluation
    switch optimization_problme
        case 'OPF'
            [Particle(i).Cost,~,~,Particle(i).Position] = test_bed_OPF...
                (test_case,scenario,trial,args,Particle(i).Position);
        case 'Numerical'
            Particle(i).Cost = numerical_Obj_Func(Particle(i).Position,func);
    end
    FEs = FEs + 1;
    % Update Personal Best
    PBest(i).Position=Particle(i).Position;
    PBest(i).Cost=Particle(i).Cost;
    % Update Global Best
    if PBest(i).Cost < GlobalBest.Cost
        GlobalBest = PBest(i);
    end
    S(i,:) = PBest(i).Position;
    Y(i) = PBest(i).Cost;
end
% kriging theta parameter
theta = 1;
% initialize the Orthogonal Array
[OA,row] = orth_array(2,nVar);
% Initialization of Award of each particle
Award = zeros(nPop,1);
% Threshold Award Values for triggering different operators
% and assigning the weight to Expected Improvement.
switch optimization_problme
    case 'OPF'
        if proc.system == 41
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 5;
        elseif proc.system == 57
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 10;
        elseif proc.system == 118
            AW_WEI = 5;
            AW_KELS = 0;
            AW_OA = 20;
        elseif proc.system == 300
            AW_WEI = 5;
```

```
            AW_KELS = 0;
            AW_OA = 30;
        end
        var_disc = ps.x_type~=0; % discrete and binary variables
    case 'Numerical'
        AW_WEI = 5;
        AW_KELS = 0;
        AW_OA = 0;
end
%% main loop
while FEs < MaxFEs
    pb = [PBest.Cost];
    for i = 1:nPop
        gb = GlobalBest.Cost;
        Candidate_Velocities = zeros(ncPop,nVar);
        % method 2
        % calculate the Candidate velocities for each particle
        for s = 1 : ncPop
            Candidate_Velocities(s,:) = w.*rand(VarSize).*Particle(i).Velocity...
                + c2*rand(VarSize).*(GlobalBest.Position - Particle(i).Position);
        end
        % Apply Velocity Limits
        Candidate_Velocities = max(Candidate_Velocities,VelMin);
        Candidate_Velocities = min(Candidate_Velocities,VelMax);

        % Update the Candidate Positions for each particle
        Candidate_Positions = Particle(i).Position + Candidate_Velocities;

        switch optimization_problme
            case 'OPF'
                % enforce the discrete and binary varibales only
                Candidate_Positions(:,var_disc) = max(Candidate_Positions...
                    (:,var_disc),VarMin(var_disc));
                Candidate_Positions(:,var_disc) = min(Candidate_Positions...
                    (:,var_disc),VarMax(var_disc));
        end
        % check whether the candidates within the specified limits, by doing so
        % only the feasible candidates remain for Kriging to select the
        % best candidate and for reducing the computational cost as well
        index = false(ncPop,1);
        for j = 1 : ncPop
            if all(Candidate_Positions(j,:) >= VarMin & Candidate_Positions(j,:) <= VarMax)
                index(j) = true; % feasible candidates
            else
                index(j) = false; % infeasible candidates
            end
        end
        if any(index)
            Candidate_Positions = Candidate_Positions(index,:);
            Candidate_Velocities = Candidate_Velocities(index,:);
            % if only one feasible candidate, no need Kriging pre-selection
            if nnz(index) == 1
                id = 1;
            else % perform Kriging pre-selection
                % use Kriging to estimate the candidates
                [YX,MSE] = kriging_predictor(S,Y,Candidate_Positions,theta);
                if Award(i) > AW_WEI
                    weight_ei = 0;
                else
                    weight_ei = 1;
                end
                % use WMEI to select the best candidate
                [~,id]= WMEI(MSE,YX,GlobalBest.Cost,weight_ei);
            end
            % update velocity and position based on the selected candidate
            Particle(i).Velocity = Candidate_Velocities(id,:);
            Particle(i).Position = Candidate_Positions(id,:);
```

```
        % evaluate the particle
        switch optimization_problme
          case 'OPF'
            [Particle(i).Cost,~,~,Particle(i).Position] = ...
              test_bed_OPF(test_case,scenario,trial,args,Particle(i).Position);
          case 'Numerical'
            Particle(i).Cost = numerical_Obj_Func(Particle(i).Position,func);
        end
        FEs = FEs + 1;
        % iteratively rolling replacement strategy for updating samples
        Y(ns+1) = Particle(i).Cost;
        S(ns+1,:) = Particle(i).Position;
        S(1,:) = [];
        Y(1,:) = [];
        % Update Personal Best
        if Particle(i).Cost < PBest(i).Cost
          PBest(i).Cost = Particle(i).Cost ;
          PBest(i).Position =  Particle(i).Position;
          % Update Global Best
          if PBest(i).Cost < GlobalBest.Cost
            GlobalBest.Cost = PBest(i).Cost;
            GlobalBest.Position = PBest(i).Position;
            Award(i) = 0; % update award
          else
            Award(i) = Award(i) + 1; % update award
          end
        else
          Award(i) = Award(i) + 1; % update award
        end
        % if all candidates are infeasible, random select one candidate
        % as the updated velocity and updated particle, but do not evaluate
        % it to save computational cost
      else
        id = randi(ncPop);
        Particle(i).Position = Candidate_Positions(id,:);
        isout = (Particle(i).Position < VarMin | Particle(i).Position > VarMax);
        if nnz(isout)~=0
          Particle(i).Velocity(isout) = Candidate_Velocities(id,isout);
        end
        Particle(i).Position = max(Particle(i).Position,VarMin);
        Particle(i).Position = min(Particle(i).Position,VarMax);
      end
      %% elstism learning
      if Award(i) > AW_KELS
        [GlobalBest,Award(i)] = KELS(Award(i),GlobalBest);
      end
      %% OA
      if PBest(i).Cost < pb(i) || GlobalBest.Cost < gb && PBest(i).Cost ~= GlobalBest.Cost
        [PBest(i),GlobalBest,Award(i)] = Kriging_OA(Award(i),PBest(i),GlobalBest);
      end
      switch optimization_problme
        case 'Numerical'
          disp(['Current Number of Function Evaluations ' num2str(FEs)...
            ': Best Cost = ' num2str(GlobalBest.Cost)]);
      end
    end
    switch optimization_problme
      case 'OPF'
        if proc.finish
          return;
        end
    end
  end
end
```

Appendix B

B1. Comparison between Real-Coded GA and Binary-Coded GA

In this thesis, the Real-Coded Genetic Algorithm (RCGA) is used because it is much more computationally efficient than the Binary-Coded GA (BCGA). Moreover, the BAGA usually encodes the variables with fixed-length strings, resulting in sacrifice of precision and the decoding procedure requiring more computation efforts than RCGA. Although it can improve the precision of the binary representation by extending more bit strings, this would considerably slow down the algorithm. Therefore, the RCGA is more suitable for this work, as it aims to improve the computational efficiency of the algorithm when solving OPF problems. Moreover, the RCGA can also easily handle the constraints of variables by directly enforcing them to their specified range.

The RCGA, BCGA and Gray-Coded GA (GCGA) were test on the numerical benchmark optimization functions. The average optimal solutions and the computation times of the 50 trials are summarized in Table B1. Due to the "Hamming Cliff" problem may occur in the BCGA causing difficult in exploration [256], thus the alternative approach – GCGA [236] – was also tested. The obtained results show that the BCGA and GCGA takes considerably longer computation time than the RCGA even through the optimal solution obtained by GCGA is overall the best. The optimal results obtained by RCGA close to the global optimum but slightly less accurate that the GCGA (except for Rastrigin function). Moreover, the BCGA obtained much worse average optima because its "Hamming Cliff" problem cannot properly explore the solution space and so the BCGA fails to find the optimum for some trails. For these reasons, the RCGA and GCGA are not considered in this work. Note that the Gray Coding (i.e. GCGA) was used to avoid the "Hamming Cliff" problem of Binary Coding and thus greatly improved the performance of BCGA but it clearly required more computational efforts.

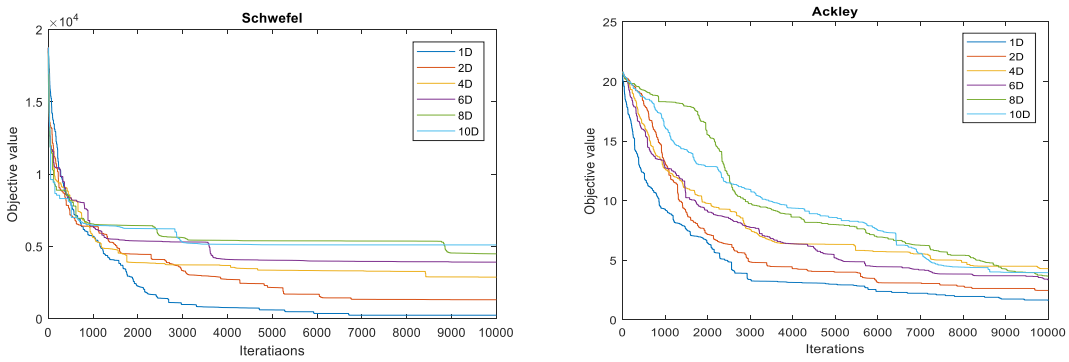
Table B1 Mean solutions and mean CPU time of the 50 trials obtained for RCGA, BCGA and GCGA

	Ackley Function		Levy Function		Rastrigin Function		Schwefel Function	
	Optimal	Time (s)	Optimal	Time (s)	Optimal	Time (s)	Optimal	Time (s)
RCG	0.0076	69.5728	1.29E-5	123.318	9.05E-4	70.9856	0.0077	78.7624
BCG	9.19E-5	1.395E+3	1.9001	1.413E+3	35.9376	1.343E+3	57.8608	1.489E+3
GCG	3.82E-4	1.4632E+	2.29E-7	1.485E+3	0.3980	1.4228E+	6.39E-4	1.707E+3

Note: the bit string length of BCGA and GCGA is calculated as: $L = round(log_2 \frac{UB-LB}{10^{-6}})$, where UB and LB are the upper bound and lower bound of the variables, respectively, 10^{-6} means the precision of six digits after the decimal point is required.

B2. Parametric Study of Elitism Learning Strategy

As discussed in previous chapters, the Gaussian based ELS operator is necessary for helping the proposed KAGA and KAPSO to avoid local optimum, since only one offspring/particle is selected to be evaluated using the real objective function while other good solutions may likely be overlooked. To further analyze this, the convergence plots of different optimization functions performed by KAGA and KAPSO are given in Fig. B3 and B4, respectively. Note, the random mutation is that the mutation offspring/particle is randomly selected (same as the mutation used in GA), whereas the elitism mutation (i.e. ELS) is that the current best solution is always selected to be mutated. It can be seen that the solutions of KAGA and KAPSO without the consideration of mutation strategy are clearly being trapped at local optima and this issue is resolved by using the random or elitism mutation strategy. Although the algorithms with the aid of different mutation strategies obtained similar final optima, the ELS performed better convergence performance (i.e. faster convergence speed). This might be caused by the randomly selected mutation offspring/particle that puts too much emphasis on the exploration, misleading the overall optimization process and resulting in slow convergence speed. It therefore suggests that the current best solution should be used to achieve better convergent performance when selecting the solution to be mutated. Furthermore, the selection of different number of mutation variables is also studied, as shown in the Fig. B1. It indicates that the smaller the number of mutation variables the better the convergence performance tends to be. Because most variables in the current best solution may actually belong to the global best solution, while selecting too many mutation variables may put extensively emphasis on exploration, leading to more thoroughly search of solution space but resulting slow convergence rate. It may be concluded that one mutation variable at a time is able to improve the computational efficiency as well as solution quality.



(a). KAPSO (b). KAGA

Fig. B1 Convergence rates of KAGA and KAPSO with different number of mutation variables

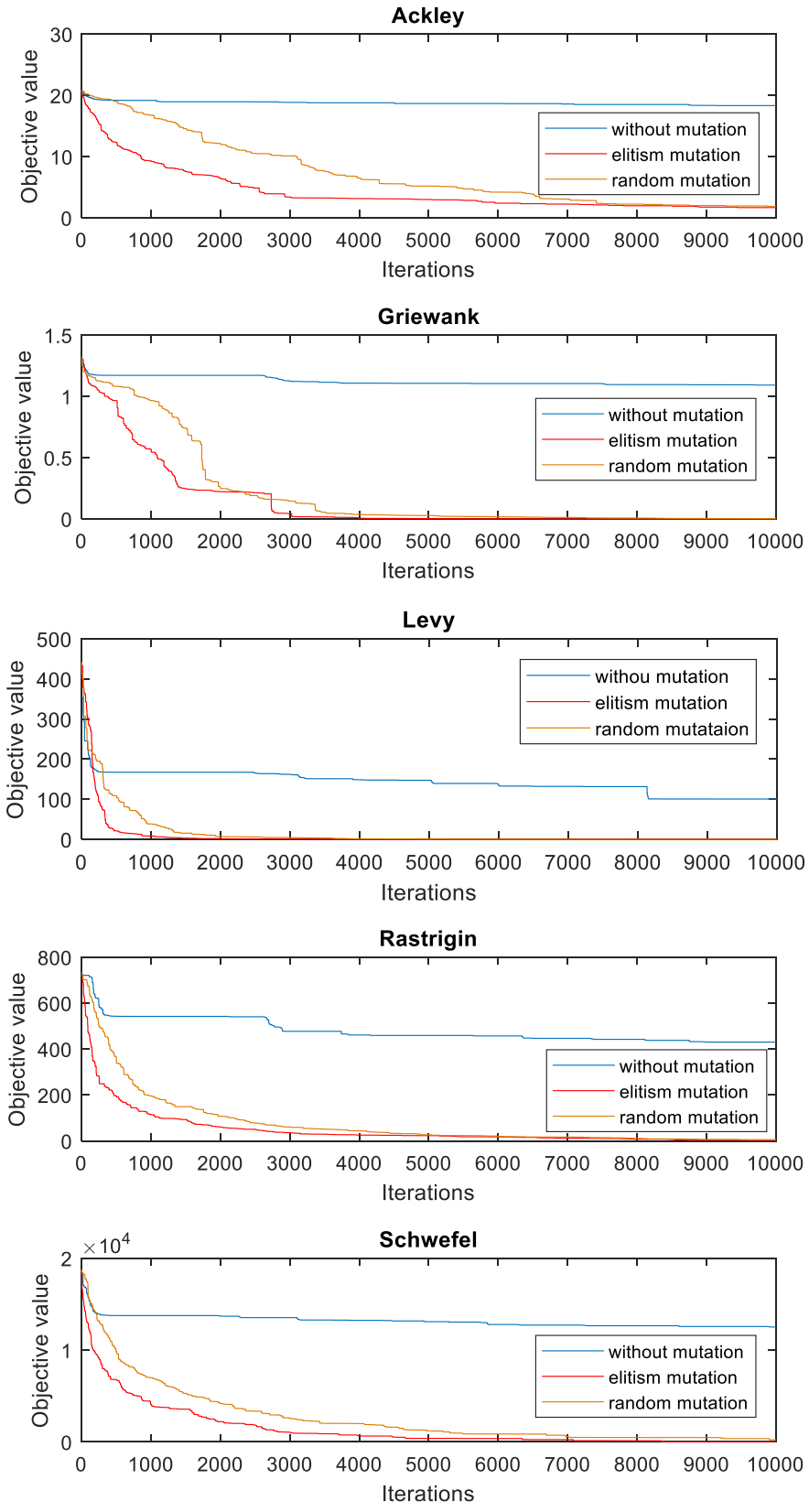


Fig. B2 Convergence performance of different functions performed by KAGA

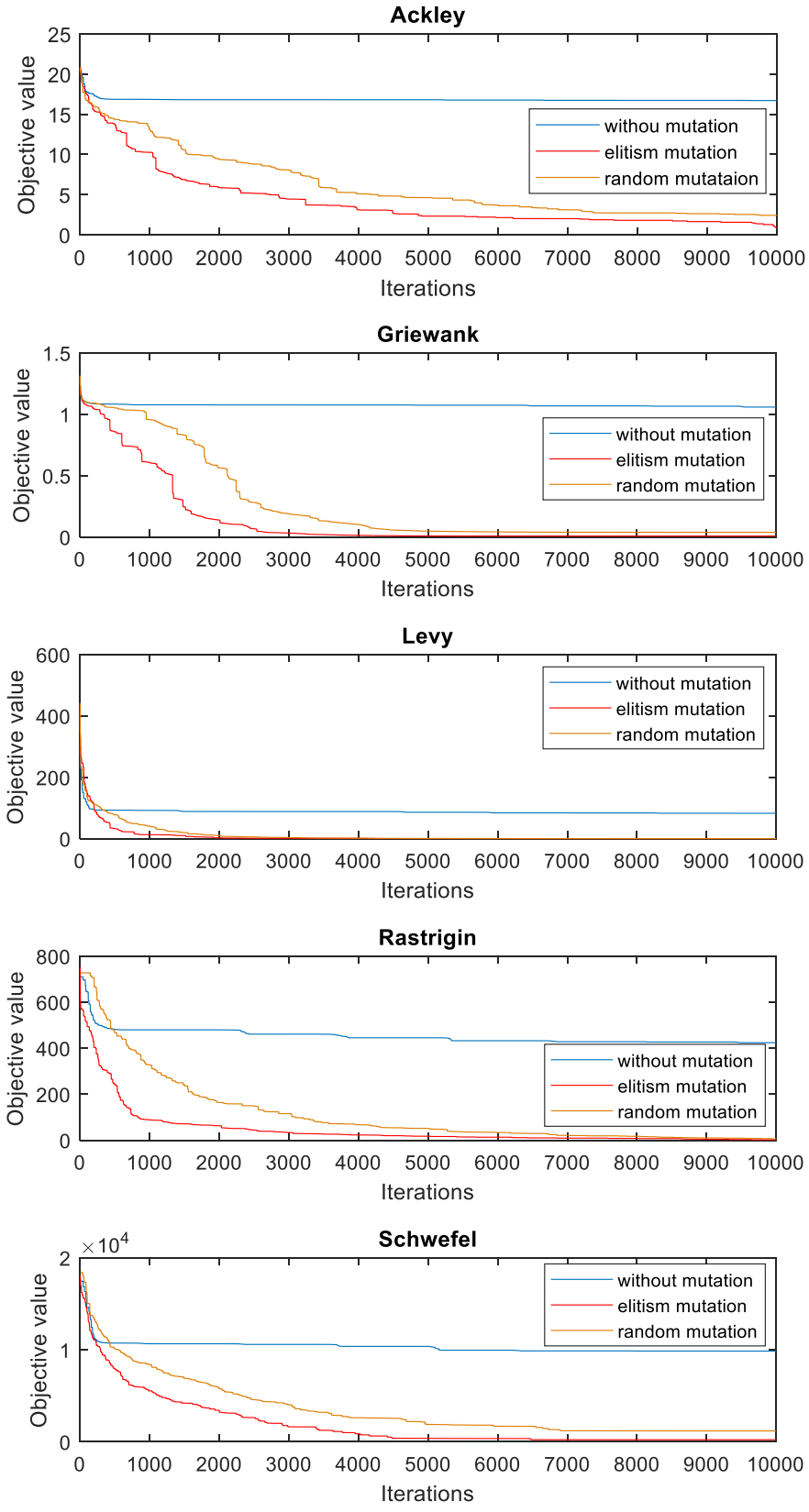


Fig. B3 Convergence performance of different functions performed by KAPSO

B3. Parametric Study of Hyperparameter in Kriging model

As introduced and discussed in chapter 3, the Kriging prediction relies highly on the correlation matrix. Moreover, the calculation of correlation values not only depends on the distance but also relates to the parameter $\boldsymbol{\theta}$ (namely hyperparameter) in (3.10). This hyperparameter is generally used to control how fast the correlation drops away from one point to another in each dimension and thus the fidelity of the model. As introduced in [208], the optimal $\widehat{\boldsymbol{\theta}}$ could be found by minimizing the function below

$$min.\ \psi(\boldsymbol{\theta}) = |\boldsymbol{R}(\boldsymbol{\theta})|\hat{\sigma}(\boldsymbol{\theta})^2 \tag{b1}$$

where (b1) is derived from (3.15), $|\boldsymbol{R}(\boldsymbol{\theta})|$ is the determinant of the correlation matrix containing pairwise distances between pairs of samples, and $\hat{\sigma}$ is the variance. The optimal $\widehat{\boldsymbol{\theta}}$ could also be considered as the maximum likelihood estimation. Normally, small $\boldsymbol{\theta}$ tends to give a correlation matrix (it consists of value one), which assumes that the solutions are highly correlated in the solution space; by contrast, large $\boldsymbol{\theta}$ tends to provide a unit correlation matrix, which assumes that the solutions are weakly correlated in the solution space. As shown in Fig. B4, the optimized $\boldsymbol{\theta}$ (obtained by using DACE toolbox) presents the most favourable and smooth Kriging model, the small $\boldsymbol{\theta}$ makes the Kriging model like a oscillating PR model, and the large $\boldsymbol{\theta}$ results in a discontinuous-like curve (the estimations away from sample points are almost identical). These results may be interpreted according to the (3.21), a unit correlation matrix or a large $\boldsymbol{\theta}$ would make the value of any estimation point away from samples to be approximately equal to the mean value of Kriging model; whereas a correlation matrix consisting of value one or a small $\boldsymbol{\theta}$ may confuse the Kriging estimation since solutions are no longer correlated (i.e. the values in the correlation matrix are all nearly equal to one).

However, the appropriate upper and lower bounds of $\boldsymbol{\theta}$ are difficult to be defined when dealing with (b1) to find the optimal $\widehat{\boldsymbol{\theta}}$. For a high-dimensional problem, the Euclidean distance between two points is generally large than the low dimensional problem. In other words, the Euclidean distance is accumulatively increased along with the dimension size, even though the dimension to dimension

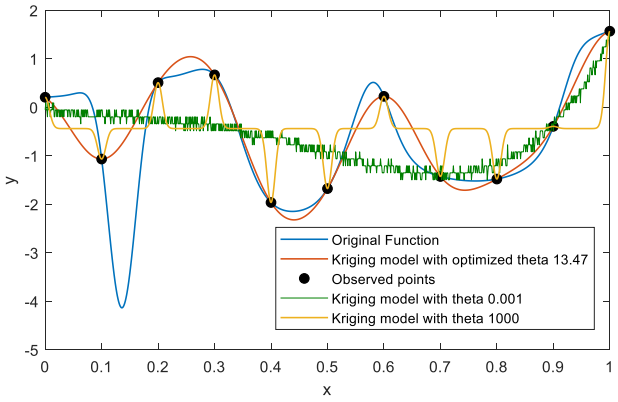


Fig. B4 Kriging model with different hyperparameter settings

value is all the same. For example, the Euclidean distance between 0.001 and 0.002 is 0.001, whereas the Euclidean distance between $\boldsymbol{a}$ (a 100-D vector consisting of value 0.001) and $\boldsymbol{b}$ (a 100-D vector consisting of value 0.002) is 0.01. This would give the high-dimensional problem a highly sparse correlation matrix and a large $\boldsymbol{\theta}$ would make it even worse; consequently, the Kriging estimations (also MSE and EI) of given points away from samples are approximately the same. As a result, the process of selecting infill point might be misled unless the global optimal $\boldsymbol{\theta}$ is found. Nevertheless, the global optimal $\boldsymbol{\theta}$ may not be easily found, it may need to set a relatively small upper bound to $\boldsymbol{\theta}$ avoid this situation happening. Furthermore, small $\boldsymbol{\theta}$ may lead to a low accuracy Kriging model, but it may not negatively affect the selection of infill point as the estimations will not be the same.

To analyse the impact of upper bound $\overline{\boldsymbol{\theta}}$ on the optimization performance, the KAGA and KAPSO with different upper bound settings were tested on five numerical functions (same as in chapter 5). Note, the script of Kriging predictor given in A3 was not used in this test, rather the DACE toolbox enabling optimization of $\boldsymbol{\theta}$ was used. Moreover, the numerical functions with true optimum zero considered 50 variables, the population size and maximum number of iterations were set to 50 and 10,000, respectively, and other related parameter settings of KAGA and KAPSO were the same as in chapter 5. The average optimal solutions of 20 trials obtained from the KAGA and KAPSO with different upper bounds $\overline{\boldsymbol{\theta}}$ are summarized in Table B2. For KAGA, the changes in $\overline{\boldsymbol{\theta}}$ caused somewhat inconclusive impacts on the optima and it may be not easy to outline the tendency, but overall the upper bound between 0.1 and 0.01may be helpful in finding better optimum. For KAPSO, it is clearer than KAGA that the smaller upper bounds produced better optima. Therefore, it is recommended to set the upper bound $\overline{\boldsymbol{\theta}}$ between 0.1 and 0.01 to improve the efficiency when using KAPSO, and it may also be potentially useful for KAGA. However, the optimization of $\boldsymbol{\theta}$ critically increases the computational cost of Kriging estimation. Therefore, the optimization of $\boldsymbol{\theta}$ was not considered in chapter 5 and chapter 6, instead, the constant values of $\boldsymbol{\theta}$ were used. It could be recommended for further work, as it could greatly improve the fidelity degree of Kriging model.

Table B2 Average optima obtained by KAGA and KAPSO with different $\overline{\boldsymbol{\theta}}$

Case	$\overline{\boldsymbol{\theta}}$	Ackley	Griewank	Levy	Rastrigin	Schwefel
			KAGA			
1	0.01	0.1785	0.0075	**0.0018**	8.1763	476.3711
2	0.1	0.2252	**0.0001**	0.0024	**0.6986**	356.6804
3	1	0.3006	0.0396	0.0019	4.7231	242.7553
4	10	0.2031	0.0079	0.0065	5.5162	119.4487
5	100	**0.1634**	0.0179	0.0066	4.2755	**119.3309**
			KAPSO			
1	0.01	0.0391	**1.11E-5**	3.83E-4	**0.0909**	237.0674
2	0.1	**0.0165**	0.0074	**2.38E-6**	2.0117	**0.0098**
3	1	0.0665	0.0246	4.35E-5	1.0289	0.0742
4	10	0.8990	0.0270	5.59E-6	0.0012	236.8915
5	100	0.4917	0.0270	6.35E-6	1.9925	118.4550

B4. The Balance Between Exploitation and Exploration When Selecting Infill Point

Because only one individual/particle in the population/swarm is selected to be evaluated using real objective function at each iteration, and the weight value is one of important element potentially impact on the selection. Therefore, the different weight strategies – including constant, linearly increasing, linearly decreasing and proposed adaptive strategy (i.e. Euclidean distance-based weight) – were studied to analyse their impacts on optimization performance of KAGA and KAPSO. The average optimal solutions of five benchmark functions are summarized in Table B3 (used the same setting as in the appendix B3). It shows that the KAGA and KAPSO using proposed weight strategy have best optimal solutions in four and three out of five functions, respectively. Although different weight values produced small differences on the average optima of Ackley, Griewank and Levy, optimal of Rastrigin and Schwefel had large differences. In reference to the solution quality, the proposed Euclidean distance-based weight strategy is reliable for improving the performance of proposed algorithms. Furthermore, it is worthy to mention that the purpose of developing Euclidean distance-based weight strategy was also to avoid the highly sparse correlation matrix or unit correlation matrix misleading the selection of infill point.

Table B3 Average optima obtained by KAGA and KAPSO with different weight values

Case	Exploitation	Exploration	Ackley	Griewank	Levy	Rastrigin	Schwefel
KAGA							
1	0.1	0.9	0.3251	0.0300	0.0543	5.0309	24.2335
2	0.3	0.7	1.1526	0.0136	0.0935	7.4869	71.6564
3	0.5	0.5	1.6523	0.0166	0.0984	8.1501	67.6564
4	0.7	0.3	2.2419	0.0140	0.1334	6.3618	56.5410
5	0.9	0.1	2.3444	0.0092	0.1412	10.0856	100.268
6	Increasing	Decreasing	1.3651	0.0135	0.1042	8.9724	40.3681
7	Decreasing	Increasing	1.6356	0.0107	0.0706	**4.3005**	164.471
8	Adapted	Adapted	**0.0907**	**0.0075**	**0.0012**	10.5097	**1.0143**
KAPSO							
1	0.1	0.9	1.2265	0.0157	0.0212	2.4251	269.2872
2	0.3	0.7	1.4569	0.0089	0.0391	3.4530	149.8779
3	0.5	0.5	1.1810	0.0176	0.0251	3.4662	374.0256
4	0.7	0.3	1.5641	5.44E-4	0.0453	4.4138	**44.3538**
5	0.9	0.1	0.9844	0.0036	0.0402	2.9237	139.9439
6	Increasing	Decreasing	0.8866	0.0154	0.0586	0.9530	154.3096
7	Decreasing	Increasing	0.9422	0.0012	0.0252	**2.2943**	268.5709
8	Adapted	Adapted	**0.8085**	**1.89E-7**	**6.56E-6**	2.9973	118.4422

B5. Comparison between Individual-based and Pre-selection Strategy on Continuous OPF

It was found that the proposed KAGA, KAPSO, PKAGA and PKAPSO algorithms were performing similarly with no distinctive difference for IEE 30 bus test cases, hence those results are not discussed and presented here. For the IEEE 118 bus test cases, the population size as well as Kriging sample size, the candidate solution size (for pre-selection), and the *MaxFEs* were set to 100, 30 and 50,000, respectively. Furthermore, as KAGA and KAPSO have been proven to outperform other reported algorithms, here the proposed pre-selection algorithms are only compared with the individual-based algorithms. Instead of using *MaxFEs* as the stopping criterion, the optimization process is terminated when the improvement on optimal solution is detected as not being updated (less than 1 for case 1 and 0.1 for case 2) for 1,000 FEs, which was considered to be an effective and robust approach to assess the convergence performance and the solution quality of the algorithms. The obtained convergent solutions of the 25 trials including mean, best, worst and std values, as well as average CPU time and FEs are given and compared in Table B4. Although the PKAGA produced more accurate solutions than KAGA for both test cases, the pre-selection Kriging used in GA required more FEs to converge and thus spent longer computation time. The PKAPSO clearly outperformed the KAPSO and others in terms of shorter elapsed time, fewer required FEs and better solution quality. Therefore, these results suggest that the proposed pre-selection is a more advanced and reliable strategy when helping stochastic algorithms to find better optimum, even though its

Table B4 Convergent Solutions, Average CPU Times and Function Evaluations of the 25 Trials for Continuous OPF

Case 1 (minimization of generation cost)				
Index	KAGA	PKAGA	KAPSO	PKAPSO
Mean ($/h)	129,698	129,632	129,628	**129,617**
Best ($/h)	129,662	129,624	129,621	**129,614**
Worst ($/h)	129,767	129,650	129,634	**129,621**
Std.	26.55	5.96	3.43	**1.75**
Avg. Time (s)	135.95	165.61	196.46	**73.49**
Avg. FEs	25,933	41,121	35,107	**16,946**
Case 2 (minimization of active power losses)				
Index	KAGA	PKAGA	KAPSO	PKAPSO
Mean (MW)	11.63	11.13	10.43	**10.35**
Best (MW)	10.90	10.40	**10.11**	10.13
Worst (MW)	13.27	12.83	10.82	**10.59**
Std.	0.49	0.53	0.16	**0.11**
Avg. Time (s)	57.94	72.70	50.89	**37.83**
Avg. FEs	12,304	15,782	10,611	**8,743**

improvement on computational efficiency sometimes may not be as good as the individual-based strategy. Moreover, comparing the KAGA and PKAGA with the KAPSO and PKAPSO reveals that the Kriging strategies applied to PSO always provides better optimum using smaller number of function evaluations, which has also been proven in the numerical benchmark tests. Furthermore, it has to be emphasized that the inconsistent solution of stochastic based algorithm is one of main issues limiting its application in practice, but here the PKAPSO is most reliable and robust due to the small std values. For example, the worst solution of the 25 trials obtained by PKAPSO for minimizing generation cost was 129,621 $/h, which was the same as the best solution given by KAPSO. It can therefore be concluded that better improvement on solution quality and computational efficiency are achieved by the proposed pre-selection strategy, while the developed KAPASO is a fast and accurate algorithm to solve continuous OPF problems.

B6. Comparison between Individual-based and Pre-selection Strategy on Mixed-Integer OPF

- *Optimal Active and Reactive Power Dispatch*

The convergent median, best and worst solutions well as the average convergent FEs and times of the 31 trials are listed in Table B5, where the results were recorded when the fitness was detected as not being updated (i.e. improvement is less than 1) for specified number of FEs (i.e. 2,000, 5,000 and 10,000 for IEEE 57, 118 and 300 bus teste system, respectively) and label "*" means that the solution is infeasible because the constraint violations are larger than 10^{-4}(i.e. since the OPF test bed only takes the violations larger than 10^{-4} into account). The convergent mean solutions are not compared here, because it was found that the calculated mean solutions are easily affected by the infeasible trials with very large penalty values. Thus directly comparing the convergent mean solutions seemed to be somewhat inconclusive, and it was decided to use the median solutions (i.e. 16th best trial) instead to make more meaningful comparisons. Regarding the overall solution quality, the individual-based and the pre-selection algorithms used shorter CPU times and offered much

Table B5 Convergent Solutions, Average CPU Times and FEs performed by proposed individual-based and pre-selection assisted algorithms (for OARPD)

IEEE 57 bus test system					
Algorithms	Median ($/h)	Best ($/h)	Worst ($/h)	Avg. FEs	Avg. Time (s)
KAGA	41,717	41,698	41,734	6,403	22.28
PKAGA	41,715	41,696	41,755	7,726	24.72
KAPSO	**41,704**	**41,691**	41,739	**4,352**	**13.05**
PKAPSO	41,708	41,691	**41,722**	5,319	16.60
DEEPSO	41,705	41,694	41,752	12,448	38.84
ICDE	42,175	41,831	45,748*	8,106	25.45
CBGA	41,749*	41,704*	41,838*	10,387	30.33
IEEE 118 bus test system					
KAGA	135,153	135,088	258,724*	22,581	198.26
PKAGA	135,200	135,095	906,469*	23,445	131.29
KAPSO	135,104	**135,021**	135,470	37,942	316.44
PKAPSO	**135,100**	135,059	**135,272**	**18,181**	**90.90**
DEEPSO	139,307	135,671	1.5E+10*	34,710	152.72
ICDE	168,835*	150,421*	6.2E+11*	26,490	114.97
CBGA	135,049*	134,969*	156,707*	57,242	245.57
IEEE 300 bus test system					
KAGA	723,923	720,496	4.9E+7*	31,090	712.41
PKAGA	722,814	720,594	1.8E+9*	52,206	523.91
KAPSO	721,346	720,500	2.7E+7*	33,235	733.67
PKAPSO	**720,580**	**720,465**	**723,997**	**28,803**	**322.76**
DEEPSO	887,939*	722,093	3.4E+12*	71,996	852.07
ICDE	742,091	734,870	898,784*	145,709	1432.7
CBGA	9.2E+13*	1.4E+13*	9.1E+15*	50,403	304.21

better generation dispatch sequence with lower cost than the ranked algorithms, particularly for large-scale power systems. For the small-scale IEEE 57 bus test system, it can be seen that the median and the best costs, as well as the average convergence times, as offered by the KAPSO, are overall the best, even if PKAPSO did obtain the lowest cost in the worst trial, although only marginally. For the IEEE 118 bus test system, the KAPSO and PKAPSO consistently found feasible solutions, while the latter provided the lowest median and worst costs as well as the shortest convergence times. Although the median and best costs of CBGA are visually the best, it was noticed that the control variables were largely beyond the specified limits. For the IEEE 300 bus test system, the PKAPSO showed its superior performance being fast and offering the lowest costs. Although the solution quality of PKAGA is not always better than KAGA, the pre-selection strategy applied to GA necessitated shorter average convergence times than the individual-based strategy and the ranked algorithms. The convergence performance performed by the proposed algorithms are given in Fig. B5.

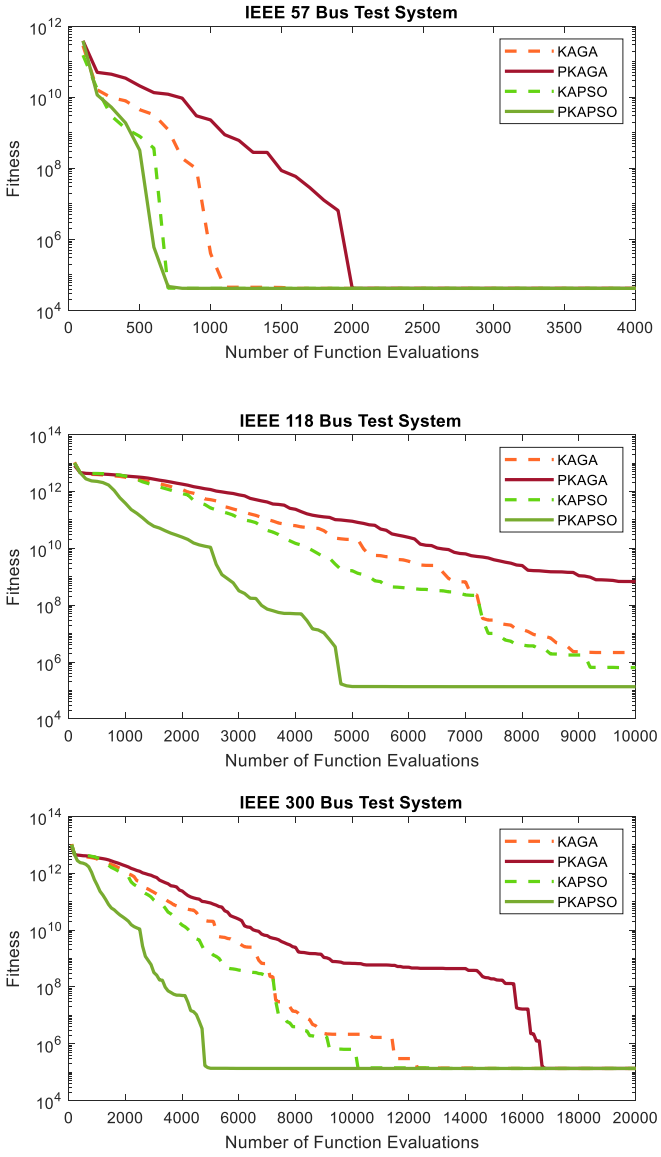


Fig. B5 Average convergence performance of 31 trials performed by proposed Kriging algorithms (for OARPD)

- *Optimal Reactive Power Dispatch*

The convergent solutions, average CPU times and number of FEs are given in Table B6. The results show that PKAPSO once again is the most reliable and robust algorithm, uses shortest computing times to provide lowest losses for all test systems, except the median and best losses obtained for the IEEE 118 bus test system which are marginally smaller as given by the ranked DEEPSO algorithm. Note also that all tested algorithms obtained infeasible solutions for all test systems for some trials. Nonetheless, the PKAPSO is by far more reliable than others as it only provided infeasible solutions with very small violations for the large-scale IEEE 300 bus test system. The developed PKAGA, although not as good as PKAPSO, did provide feasible solutions and is more efficient than the ranked algorithms. Furthermore, the results of the individual-based and the pre-selection algorithms are comparable; however, the algorithms assisted by the pre-selection Kriging used shorter CPU times to converge. This also demonstrates that the pre-selection is a more efficient strategy than the individual-based strategy. For a better view, the convergence performance performed by the proposed algorithms are given in Fig. B6.

Table B6 Convergent Solutions, Average CPU Times and FEs performed by proposed individual-based and pre-selection assisted algorithms (for ORPD)

IEEE 57 bus test system					
Algorithms	Median (MW)	Best (MW)	Worst (MW)	Avg. FEs	Avg. Time (s)
KAGA	25.73	24.79	77.95*	3,541	12.82
PKAGA	25.29	24.78	1.1E+4*	4,500	15.12
KAPSO	25.00	24.67	26.23	**2,967**	**9.67**
PKAPSO	**24.86**	**24.65**	**25.38**	3,229	11.04
DEEPSO	27.17	25.82	6.2E+6*	5,922	19.66
ICDE	27.91	25.77	35.57*	6,993	22.38
CBGA	27.68*	26.44*	29.21*	4,035	12.19
IEEE 118 bus test system					
KAGA	125.41	120.21	1,915.37*	12,745	99.28
PKAGA	127.11	121.60	6,200.36*	15,180	76.51
KAPSO	123.19	118.77	1,396.52*	12,967	95.05
PKAPSO	122.93	119.54	**128.50**	**11,651**	**51.85**
DEEPSO	**122.20**	**119.51**	1.5E+10*	29,464	123.46
ICDE	139.72	129.66	13,642.3*	25,983	106.01
CBGA	129.27*	117.10*	144.16*	18,119	77.73
IEEE 300 bus test system					
KAGA	542.20*	398.04	6,149.75*	31,090	676.01
PKAGA	456.91*	393.21	6,520.90*	52,206	561.74
KAPSO	593.13*	391.05	6,755.04*	33,235	667.92
PKAPSO	**388.53**	**385.63**	**748.95***	**28,803**	**284.29**
DEEPSO	408.34	397.52	6,446.34*	71,996	660.93
ICDE	7,684.47*	400.09	5.0E+9*	145,709	1305.56
CBGA	391.54*	**375.40***	3,485.03*	50,403	453.97

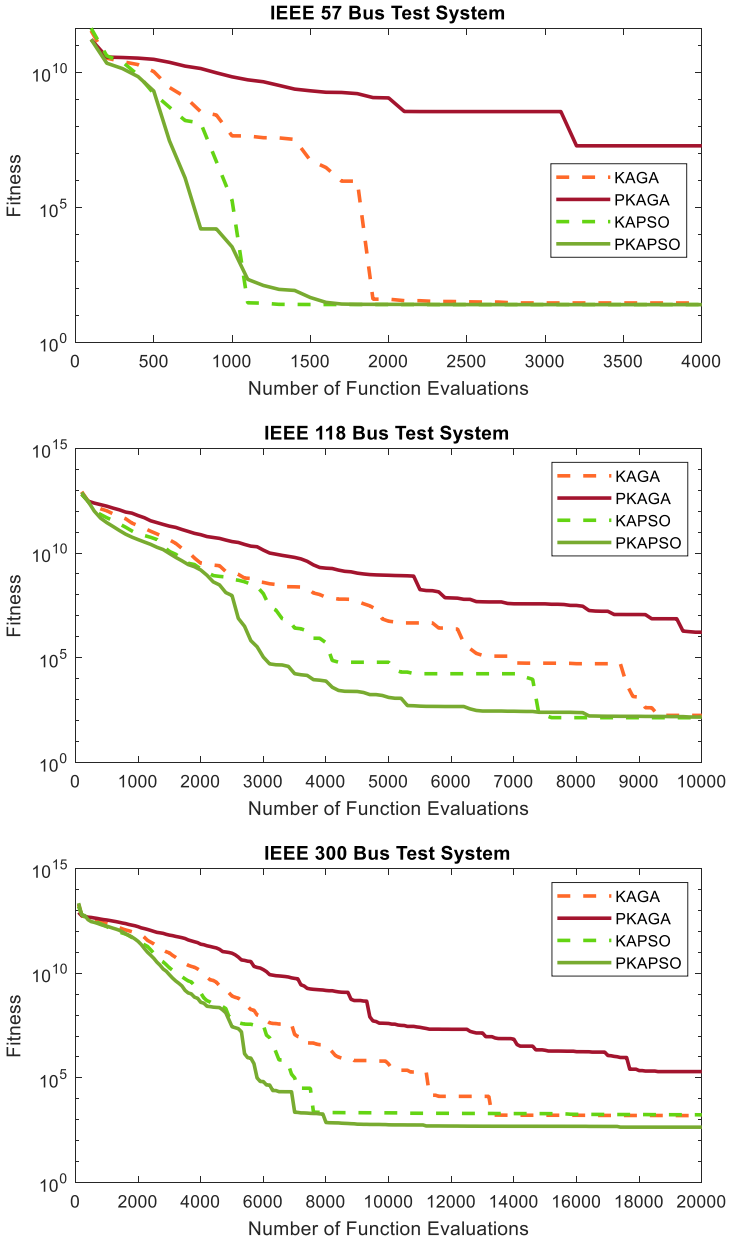


Fig. B6 Average convergence performance of 31 trials performed by proposed Kriging algorithms (for ORPD)

B7. Parametric Study of Pre-selection Kriging with Different Number of Candidate solutions

To analyse the impact of candidate solution size (nc) on the computation performance of pre-selection Kriging assisted algorithms, the PKAGA and PKAPSO with different settings of nc were tested on OARPD and ORPD problems. The convergent median, mean, best, worst and average FEs are given in Table B7 and B8, while the convergence plots are illustrated in Fig. B7-B10. It may not be easy to make the conclusive answer for which candidate size is the best according to these simulations results, because different nc caused minor differences. Sometimes small nc produced better convergence performance in some test cases but worse in other cases, similar to large nc. Nevertheless, it was noticed that the larger value of nc produced slightly better optimum in most cases, whereas smaller value of nc required relatively smaller number of FEs and converged faster. Moreover, from the practical point view, the pre-selection Kriging strategy may be a robust technique because the PKAGA and PKAPSO produced similar results when the candidate size was changed. Therefore, the user no need to focus on tuning the candidate size and to worry it would make significant impact on the algorithm's computational performance.

- *Optimal Active and Reactive Power Dispatch*

Table B7 Convergent Solutions, Average CPU Times and FEs performed by pre-selection Kriging algorithms when using different number of candidate solutions (for OARPD)

IEEE 57 bus test system					
Algorithm (nc)	Median	Mean ($/h)	Best ($/h)	Worst ($/h)	Avg. FEs
PKAGA (1)	41,715	41,720	**41,694**	41,767	**7,174**
PKAGA (20)	**41,715**	**41,717**	41,696	**41,755**	7,726
KAPSO (30)	41,719	41,723	41,697	41,783	7,465
PKAPSO (1)	41,708	41,709	**41,690**	41,728	5,329
PKAPSO (20)	41,708	41,708	41,691	**41,722**	5,319
PKAPSO (30)	**41,704**	**41,704**	41,691	41,727	**5,294**
IEEE 118 bus test system					
PKAGA (1)	136,491	2.0E+06	135,133	6.0E+07	**18,023**
PKAGA (30)	135,200	160,120	135,095	906,469	23,445
KAPSO (100)	**135,171**	**135,197**	**135,082**	**135,626**	30,981
PKAPSO (1)	**135,069**	**135,088**	**135,018**	135,274	21,294
PKAPSO (30)	135,100	135,120	135,059	**135,272**	**18,181**
PKAPSO (100)	135,094	135,106	135,035	135,300	19,090
IEEE 300 bus test system					
PKAGA (1)	722,292	1.0E+07	720,619	3.0E+08	**34,081**
PKAGA (50)	722,814	1.0E+07	720,594	**2.0E+08**	41,768
KAPSO (150)	**722,161**	1.0E+07	**720,533**	4.0E+08	57,171
PKAPSO (1)	720,587	1.0E+06	720,493	2.0E+07	**24,152**
PKAPSO (50)	**720,580**	**721,234**	720,465	**723,997**	29,684
PKAPSO (150)	720,601	1.0E+06	**720,460**	2.0E+07	25,890

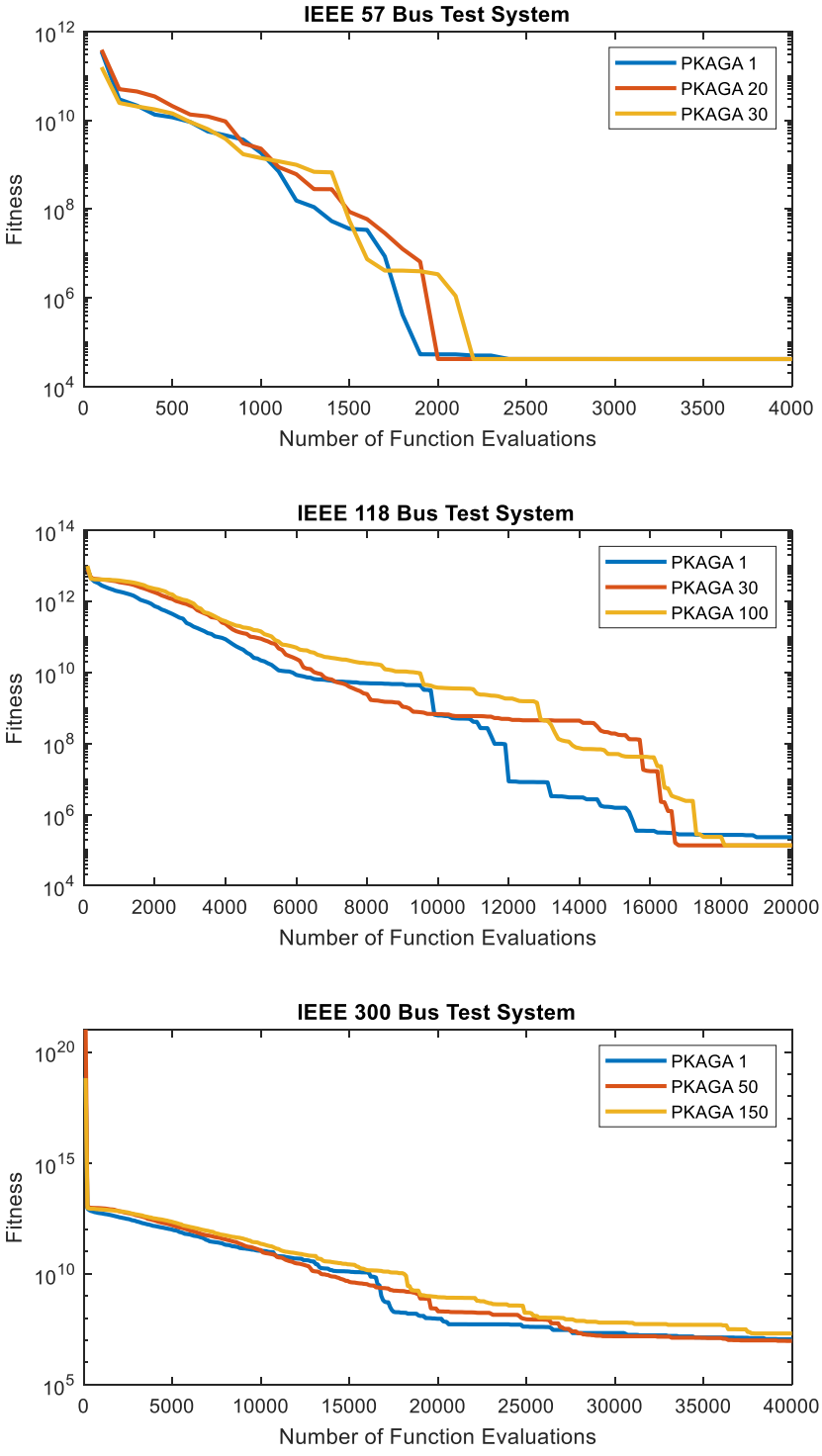


Fig. B7 Average convergence performance of 31 trials performed by PKAGA using different sizes of candidate solutions (for OARPD)

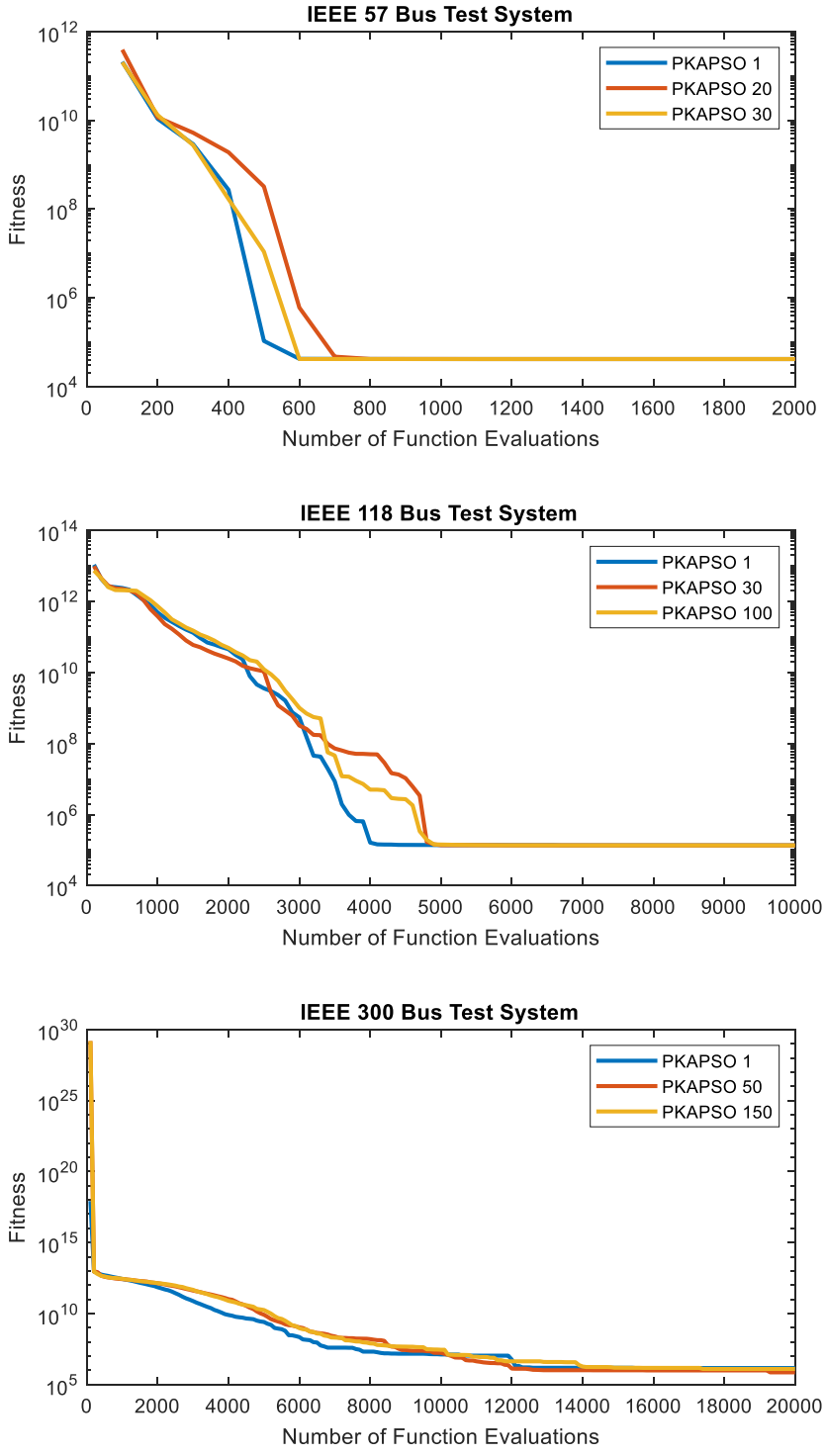


Fig. B8 Average convergence performance of 31 trials performed by PKAPSO using different sizes of candidate solutions (for OARPD)

- *Optimal Reactive Power Dispatch*

Table B8 Convergent Solutions, Average CPU Times and FEs performed by pre-selection Kriging algorithms when using different number of candidate solutions (for ORPD)

Algorithms (*nc*)	Median ($/h)	Mean	Best ($/h)	Worst ($/h)	Avg. FEs
IEEE 57 bus test system					
PKAGA (1)	25.81	**26.35**	24.81	39.43	**4,216**
PKAGA (20)	**25.29**	389.69	24.78	1.10E+04	4,500
KAPSO (30)	25.69	180.40	**24.77**	4.45E+03	4,242
PKAPSO (1)	24.94	25.05	24.62	25.89	**3,065**
PKAPSO (20)	24.86	**24.93**	**24.65**	**25.38**	3,229
PKAPSO (30)	**24.85**	24.99	**24.65**	25.84	3,187
IEEE 118 bus test system					
PKAGA (1)	1,130.24	3,4361.26	124.54	7.64E+05	**14,410**
PKAGA (30)	127.11	**483.11**	121.60	**6.20E+03**	15,181
KAPSO (100)	**125.99**	614.43	**120.51**	1.52E+04	15,261
PKAPSO (1)	123.02	**123.17**	119.12	**127.58**	11,194
PKAPSO (30)	**122.93**	123.19	119.54	128.50	11,652
PKAPSO (100)	123.29	124.27	**118.68**	130.74	**10,803**
IEEE 300 bus test system					
PKAGA (1)	653.45	692,753	399.09	1.28E+07	**46,381**
PKAGA (50)	456.91	1,549.67	**393.21**	6.52E+03	52,206
KAPSO (150)	**418.13**	**1,347.04**	406.77	7.16E+03	54,571
PKAPSO (1)	494.37	1,016.65	392.64	3.05E+03	29,577
PKAPSO (50)	**388.53**	**438.53**	**385.63**	**7.49E+02**	**28,803**
PKAPSO (150)	404.61	897.99	393.00	4.18E+03	29,329

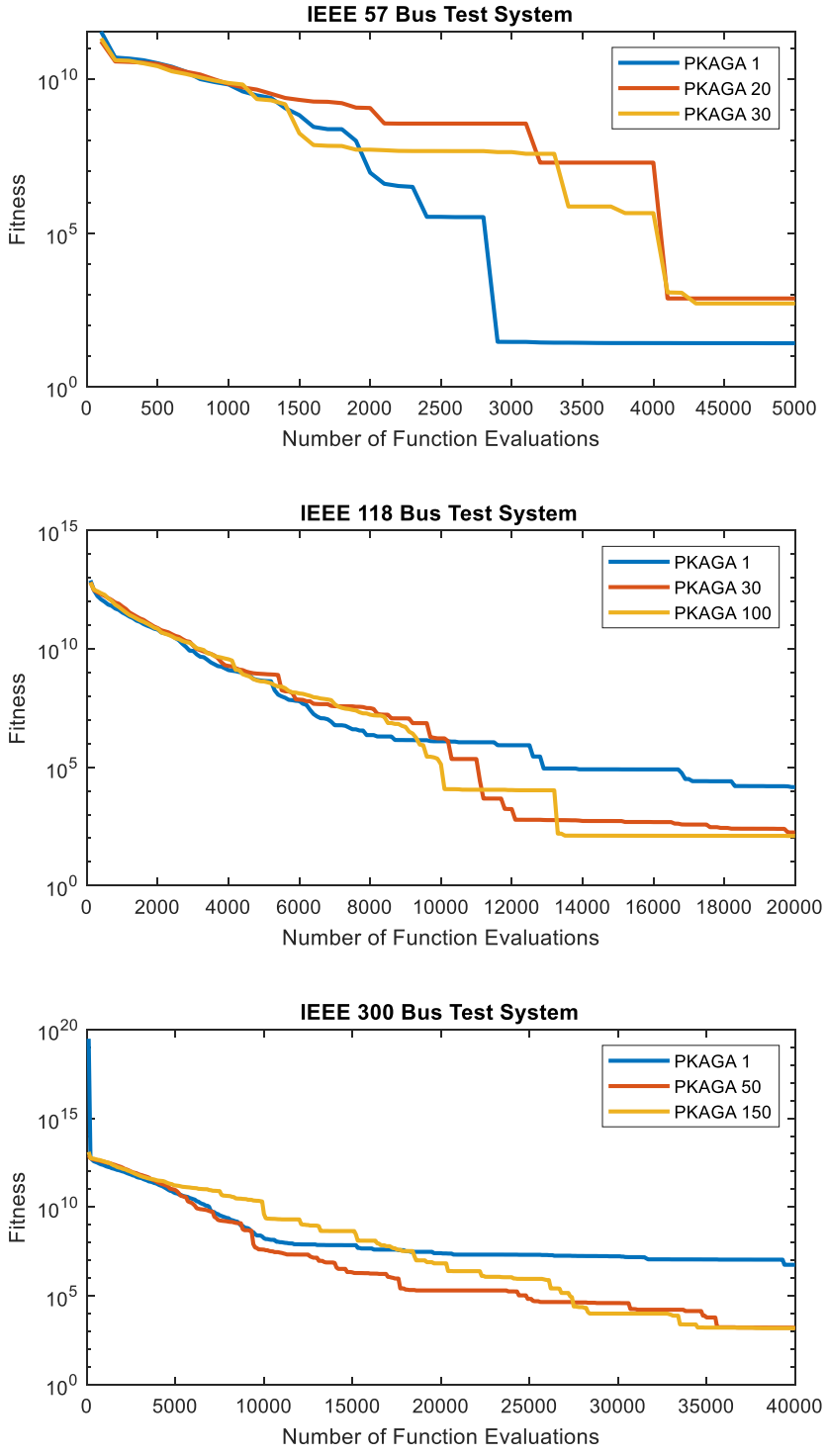


Fig. B9 Average convergence performance of 31 trials performed by PKAGA using different sizes of candidate solutions (for ORPD)

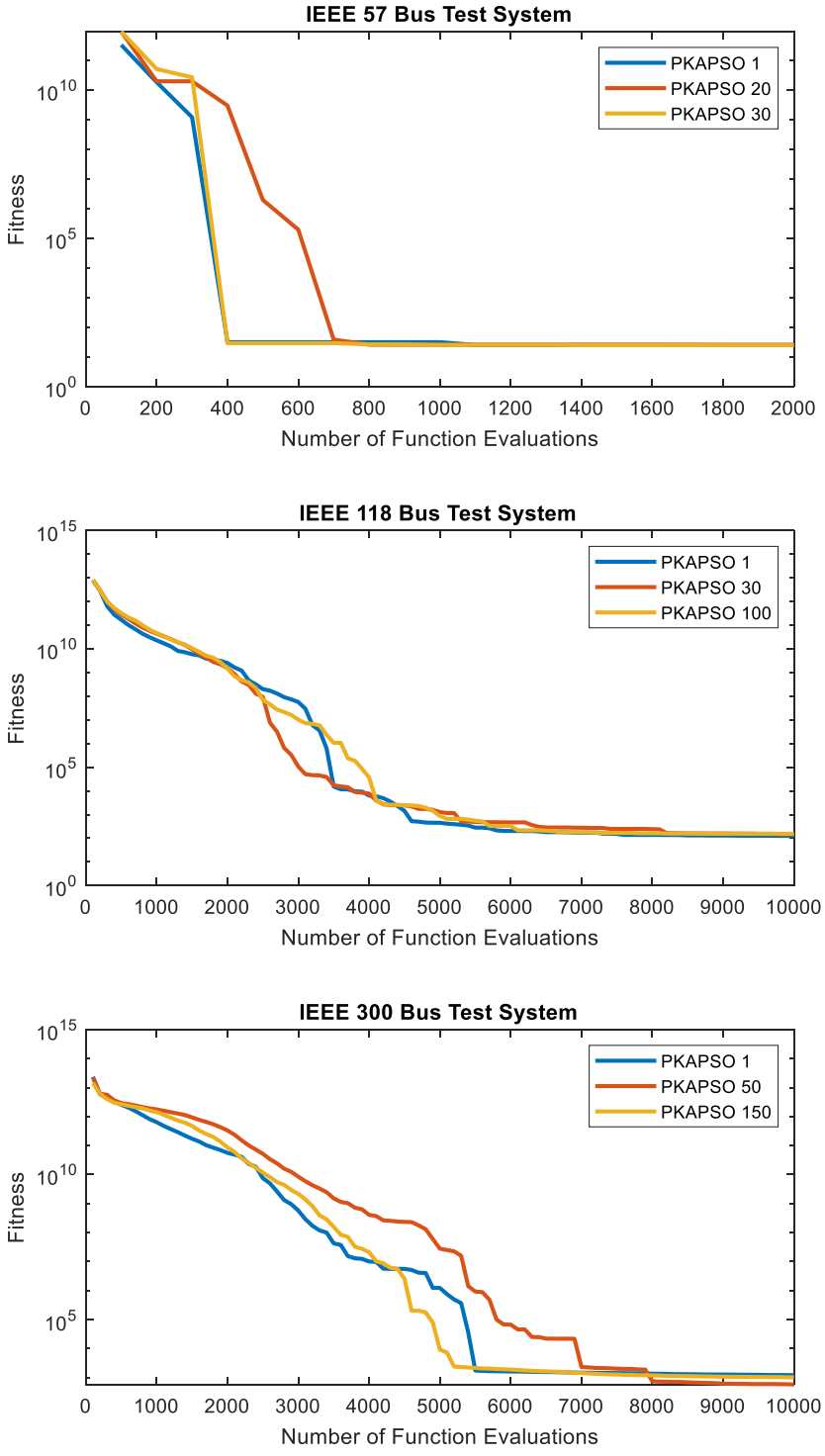


Fig. B10 Average convergence performance of 31 trials performed by PKAPSO using different sizes of candidate solutions (for ORPD)