Kriging-assisted evolution strategy for optimization and application in material parameters identification

Changwu Huang

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité Mécanique

Préparée au sein de L'Institut National des Sciences Appliquées Rouen Normandie

Contribution à l'optimisation évolutionnaire assistée par modèle de Krigeage: Application à l'identification des paramètres en mécanique

Kriging-Assisted Evolution Strategy for Optimization and Application in Material Parameters Identification

Présentée et soutenue par
Changwu HUANG

Thèse soutenue publiquement le 06 Avril 2017 devant le jury composé de		
Thierry BARRIERE	Professeur des Universités - Institut FEMTO-ST Université de Franche-Comté	Rapporteur
Abel CHEROUAT	Professeur des Universités - Université de Technologie de Troyes	Rapporteur
Pierre Richard DAHOO	Professeur des Universités - Université de Versailles Saint-Quentin-en-Yvelines	Examinateur
Philippe POUGNET	Docteur - Valeo Seimens eAutomotive France SAS-R&D Department.	Examinateur
Tarek MERZOUKI	Maître de conférences - Université de Versailles Saint-Quentin-en-Yvelines	Examinateur
Abdelghani SAOUAB	Professeur des Universités - Université du Havre	Examinateur
Bouchaïb RADI	Professeur des Universités - FST Settat, Maroc	Examinateur
Abdelkhalak EL HAMI	Professeur des Universités - INSA Rouen Normandie	Directeur de Thèse

Thèse dirigée par Abdelkhalak EL HAMI, Laboratoire de Mécanique de Normandie (LMN)

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

ED PSIME

LOFTMS

Acknowledgments

First and foremost, I would express my sincere acknowledgment to China Scholarship Council (CSC) for its financial support of my study in France and to my supervisor Professor Abdelkhalak EL HAMI for his guidance, patience, encouragement and providing me the opportunity to do research on such an interesting and rewarding subject. I have greatly benefited from the supervision style of my supervisor which enhances my ability to think independently and so that I can carry out research independently in the future.

I would like to express my appreciation to Professor Boucha ƀ RADI for his careful guidance and valuable advice during our collaboration. I am very grateful for the period of time when we worked together and discussed about our research papers.

I would also like to express my sincere gratitude to all the members of my doctoral thesis committee for their insightful comments on my thesis and encouragement.

To all my colleagues in LMN and my friends who study in INSA de Rouen and CORIA, Mayssam Al Soufi , Hao BAI, Yujun CAO, Zelong MA, ……, thank you very much for your valuable supports and aids in my study and life.

I wish to express special thanks to my girlfriend, who is always with me although we are separated by ten thousand kilometers, for her companion and love which make even ordinary days romantic and meaningful.

Lastly, I offer sincere thanks to all my family members, for their encouragement and faith in me.

Abstract

Optimization is widely required and applied in science and in engineering. Many powerful optimization techniques (algorithms) have been developed for different kinds of optimization problems. An emerging class of challenging optimization problems is known as expensive optimization problems. The high computational cost of solving this kind of problems can arise due to the high expense in objective function evaluations, unavailability of derivative information, and complex landscape of objective function. In order to reduce the cost of solving expensive problems, this thesis devoted to Kriging-Assisted Covariance Matrix Adaptation Evolution Strategy (KA-CMA-ES).

In this thesis, several propositions for the open questions in surrogate-assisted evolutionary optimization have been developed and applied in KA-CMA-ES. The developed KA-CMA-ES algorithms were analyzed and evaluated by experiments on forty test problems. Applications of the proposed KA-CMA-ES algorithm were carried out in material parameter identification of an elastic-plastic damage constitutive model.

The results of experimental studies demonstrate that the developed KA-CMA-ES algorithms are more efficient than the standard CMA-ES and that the KA-CMA-ES using modified approximate ranking procedure with Expected Improvement as metric (ARP-EI) has the best performance among all the investigated KA-CMA-ES algorithms in this work. The results of engineering applications of the algorithm ARP-EI in inverse method of material parameter identification show that the presented elastic-plastic damage model is adequate to describe the plastic and ductile damage behavior of the used material and also prove that the proposed KA-CMA-ES apparently improve the efficiency of the standard CMA-ES. Therefore, the KA-CMA-ES is more powerful and efficient than CMA-ES for expensive optimization problems.

Keywords: expensive optimization problem, evolution strategy, CMA-ES, Kriging model, pre-selection, evolution control, approximate ranking procedure, parameter identification.

Résumé

L'optimisation est une problématique largement demandée et appliquée en science et en ingénierie. Beaucoup d'algorithme d'optimisation sont développées pour les différents types de problèmes dont un type classique et connu est le Problème d'Optimisation Couteuse. Les coûts pour résoudre ce type de problème peut se produire en raison de la dépense dans l'évaluations de la fonction d'objectif, de l'indisponibilité de fonction dérivée et du contexte complexe pour la fonction objective. Afin de réduire le coût de calcul pour des Problèmes d'Optimisation Coûteuse, cette thèse a été consacrée à la Stratégie d'Evolution avec Adaptation de Matrice de Covariance assistée par modèle de Krigeage (KA-CMA-ES).

Dans cette thèse, plusieurs propositions pour les questions ouvertes dans l'optimisation évolutive assistée par substitution sont développées et appliquées dans KA-CMA-ES. Les algorithmes KA-CMA-ES développés sont analysés et évalués par 40 cas-test. Les applications de l'algorithme KA-CMA-ES développé sont réalisées par l'identification des paramètres matériels avec un modèle constitutif d'endommagement élastoplastique.

Les résultats expérimentaux démontrent que les algorithmes KA-CMA-ES développés sont plus efficaces que le CMA-ES standard. Ils justifient également que le KA-CMA-ES couplé avec ARP-EI est le plus performant algorithme par rapport aux autres méthodes étudiées dans ce travail. Les résultats obtenus par l'algorithme ARP-EI dans l'identification des paramètres matériels montrent que le modèle d'endommagement élastoplastique utilisé est satisfait pour décrire le comportement d'endommage plastique et ductile. D'ailleurs, ils prouvent que le KA-CMA-ES proposée améliore l'efficace de la CMA-ES. Par conséquent, le KA-CMA-ES est plus puissant et efficace que CMA-ES pour des Problèmes d'Optimisation Coûteuse.

Mots-clés: problème d'optimisation coûteuse, stratégie d'évolution, CMA-ES, modèle de Krigeage, présélection, contrôle d'évolution, procédure de classement approximatif, identification des paramètres.

Publications

The papers that were produced during my PhD study are listed below:

Huang, C., Radi, B., & Hami, A. E. (2016). Uncertainty analysis of deep drawing using surrogate model based probabilistic method. The International Journal of Advanced Manufacturing Technology, 86(9-12), 3229–3240. doi:10.1007/s00170-016-8436-4

Huang, C., El Hami, A., & Radi, B. (2016). Metamodel-based inverse method for parameter identification: elastic–plastic damage model. Engineering Optimization, 49(4), 633–653. doi:10.1080/0305215x.2016.1206537

Huang, C., El Hami, A., & Radi, B. (2017). Overview of Structural Reliability Analysis Methods — Part I : Local Reliability Methods. Incertitudes et Fiabilité Des Systèmes Multiphysiques, 17(1). doi:10.21494/iste.op.2017.0115

Huang, C., El Hami, A., & Radi, B. (2017). Overview of Structural Reliability Analysis Methods — Part II : Sampling Methods. Incertitudes et Fiabilité Des Systèmes Multiphysiques, 17(1). doi:10.21494/iste.op.2017.0116

Huang, C., El Hami, A., & Radi, B. (2017). Overview of Structural Reliability Analysis Methods — Part III : Global Reliability Methods. Incertitudes et Fiabilité Des Systèmes Multiphysiques, 17(1). doi:10.21494/iste.op.2017.0117

Contents

List of Figures

List of Tables

List of Algorithms

General Introduction

Optimization is extensively required and applied in almost all disciplines, whether economics, sciences, or engineering. Commonly, in engineering design and management, we almost always try to optimize something, for instance, to minimize the cost and energy consumption, or to maximize the profit, performance and efficiency. In real life, resources, time and money are always limited; consequently, optimization is far more important. Motivated by industrial and research demands, many powerful optimization techniques (algorithms) have been developed. The algorithm chosen for an optimization task largely depend on the type of the problem, the nature of the algorithm, the desired quality of solution, the available computing resource, time limit, availability of the algorithm implementation, and the expertise of the decision-makers [1]. This thesis devotes to a class of challenging optimization problems known as expensive optimization problems [2]. In this introductory chapter, the scope and motivation of our study are firstly explained. Then, we give a brief outline of the whole thesis.

Aim and Scope

With the development of numerical methods and computing technology, contemporary engineering design is heavily based on computer simulations, such as finite element method (FEM) and computational fluid dynamics (CFD) simulations. Correspondingly, computer-aided design optimization is now involved in a wide range of applications, for instance, optimization of automotive components and shape optimization of wind turbine blades. However, computer-aided design optimization is accompanied by several difficulties, which bring high computational cost in optimization. Thus, computer-aided design optimization generally belongs to the so-called expensive optimization problems. The computational challenge arises in this class of expensive optimization problems generally are due to:

- Objective function is evaluated based on computer simulation, which can be expensive (require anywhere from minutes to hours even days of computation time for each run of simulation). Therefore, in this case, the computational cost would be very expensive, because usually a large number of objective function evaluations is needed in the optimization process.
- There is no explicit analytical expression for the objective function or its derivatives. Thus, derivative-free algorithms are required to solve this black-box type of optimization problem. Generally, without the aid of derivative information, derivative-free algorithms require more evaluations of objective function than derivative-based algorithms.
- For many computer-aided design optimization problems in industry and engineering, the landscape of objective function may be non-smooth, multimodal, discontinuous and ill-conditioned. These difficulties also bring about high computational cost in finding the optimum.

In this thesis, our objective is to develop powerful optimization techniques that can efficiently deal with the expensive optimization problems subject to the box constraints. This kind of problems can be described as

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{l} \le \mathbf{x} \le \mathbf{u} \end{aligned}$$

where $f(\mathbf{x}): \mathbb{R}^d \mapsto \mathbb{R}$ is the objective function which is evaluated by running computer simulation, vector $\mathbf{x} = [x_1, \cdots, x_d]^{\mathrm{T}}$ is the collection of d design variables, vectors $\mathbf{l}$ and $\mathbf{u}$ are called lower and upper bounds, respectively. Box constraints restrict the search space to the hyperrectangle $[l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d]$.

Because of the black-box property and complex landscape of objective function in expensive optimization problems, evolutionary algorithms (EAs), which is a class of derivative-free and powerful global optimizers, are appropriate for solving expensive optimization problems. However, the main difficulty in applying EAs to solve expensive optimization problems is that EAs usually need a large number of fitness function (objective

function) evaluations before obtaining a satisfying result. Consequently, surrogate-assisted evolution algorithms were motivated from reducing computational costs in evolutionary optimization of expensive problems. Optimization using surrogate-assisted evolutionary algorithms is also known as surrogate-assisted evolutionary optimization or surrogate-assisted evolutionary computation in the optimization community.

This work concentrates on surrogate-assisted evolution strategy (ES) for expensive optimization problems. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Kriging model are chosen as the two components of surrogate-assisted evolution strategy. This Kriging-Assisted Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is abbreviated as KA-CMA-ES in the thesis. Our goal is to investigate the existed surrogate-assisted ES and develop new efficient algorithms of KA-CMA-ES for expensive problems.

A comprehensive study on KA-CMA-ES are performed in this thesis. New training set selection methods, pre-selection strategy, evolution control and approximate ranking procedure have been modified or developed, and the corresponding KA-CMA-ES algorithms are formulated. Then, experimental studies of KA-CMA-ES algorithms using existed training set select methods, pre-selection and evolution control, and our new developed KA-CMA-ES algorithms are carried out to analyze and evaluate the performance of the algorithms.

Thesis Outline

This thesis is organized in six chapters. After this general introduction, Chapter 1 gives the state of the art of optimization. Some basic concepts of optimization and a brief overview of optimization algorithms are firstly presented in Chapter 1. Then, a review of surrogate-assisted evolutionary computation based fitness approximation is offered.

Chapter 2 systematically presents the evolution strategies (ESs), from the main principles and evolutionary operators to strategy parameter control and algorithms. The

CMA-ES, which is one of the most successful evolution strategies and is chosen as the representative of ESs, is completely introduced.

Chapter 3 introduces the surrogate modeling of computer experiments. Design of experiments, surrogate models (including polynomial regression model, Kriging model, radial basis function model, multi-layer perceptron networks and support vector regression), and model validation are successively described in this chapter.

Chapter 4 dedicates to the Kriging-Assisted CMA-ES (KA-CMA-ES) algorithms, which combine the Kriging model and CMA-ES to reduce the computational cost of ES for expensive optimization problems. A brief survey of surrogate-assisted evolution strategies and some open issues in this topic are firstly provided. We focus on dealing with these open questions and developing new KA-CMA-ES algorithms for expensive optimization problems. Then, a series of KA-CMA-ES algorithms using pre-selection strategy, individual-based control, approximate ranking procedure and generation-based control are illustrated in mathematical formulations. Finally, experimental studies on KA-CMA-ES algorithms are performed on forty test problems (including twelve benchmarking functions), to analyze and evaluate the performance of existed and our newly developed algorithms.

Chapter 5 provides an engineering application of the KA-CMA-ES algorithm using approximate ranking procedure in material parameter identification of an elastic-plastic damage model.

Chapter 6 gives a brief summary of the whole thesis in French.

Lastly, the conclusion and perspective part concludes the thesis and summarizes our contributions to surrogate-assisted evolution strategies for expensive optimization problems.

1. State of the Art of Optimization

This chapter presents a brief overview of optimization techniques and the state of the art of surrogate-assisted evolutionary optimization. Firstly, basic concepts of optimization are concisely introduced in Section 1.1. An overview of optimization algorithms is given subsequently in Section 1.2, including derivative-based and derivative-free algorithms. Section 1.3 reviews the development of surrogate-assisted evolutionary algorithms (EAs) for optimization.

1.1 Basic Concepts of Optimization

Optimization means finding the best solution, which is specified by certain goals, such as minimizing the cost of a process or maximizing the efficiency of a system from all feasible solutions [1]. Feasible solutions are those that satisfy all the conditions or constraints in the optimization problem. Nowadays, optimization is required and applied in almost all disciplines, whether economics, science, or engineering. In this section, some basic notions of optimization are presented.

In an optimization problem, a function that mathematically represents the problem is to be minimized or maximized. The function that is being optimized is referred to as the objective function. Generally, the objective function is a quantity such as cost, profit, efficiency, size, weights, output, and so on. The variables in the objective function are denoted as the design variables. For example, for a structural optimization problem, design variables could be the dimensions of a structure or its material parameters. In some problems, design variables take on values from a discrete set, often a subset of integers, whereas in other problems, design variables can take on any real values. Problems with discrete design variables are called discrete optimization problems; similarly, problems with continuous variables are continuous optimization problems. In some optimization problems, there are some constraints on design variables, which restrict the domain from which design variables

can be taken values. In other words, the values that design variables taken should satisfy the constraints, if constraints exist. In this work, only the continuous optimization problems with box constraints are considered.

The standard mathematical form of an (continuous) optimization problem is

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && g_i(\mathbf{x}) \le 0, \quad i = 1, \cdots, m \\ &&& h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, p \end{aligned} \tag{1.1}$$

where $f(\mathbf{x}): \mathbb{R}^d \mapsto \mathbb{R}$ is the objective function to be minimized over the design variables that are collected in a vector $\mathbf{x} = [x_1, \cdots, x_d]^{\mathrm{T}}$, $g_i(\mathbf{x}) \le 0$ are called inequality constraints, and $h_j(\mathbf{x}) = 0$ are called equality constraints. Conventionally, the standard form defines a minimization problem. A maximization problem can be treated by negating the objective function. The constraints on design variables define the set or domain

$$S = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, g_i(\mathbf{x}) \le 0, \quad i = 1, \cdots, m \text{ and } h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, p \right\}, \tag{1.2}$$

which is called feasible set (also called feasible region) or search domain, and any element from this set is called a feasible point. A feasible point $\mathbf{x}^*$ is called global minimum (global optimum) solution, if

$$f^* = f(\mathbf{x}^*) \le f(\mathbf{x}) \text{ for all } \mathbf{x} \in S. \tag{1.3}$$

Conversely, it is called a local minimum (local optimum) if the above inequality holds for $\mathbf{x}$ only within its neighborhood [3]. The global and local optimum are simply illustrated in Figure 1.1.

The optimization problem described in formulation (1.1) is called a constrained optimization problem since there are constraints on design variables. Optimization problem without constraints are correspondingly referred to as unconstrained optimization problems. Unconstrained optimization problems arise directly in many practical applications. They are also the basis for constrained optimization problems in which the constraints can be replaced by a penalty term in the objective function. Constrained optimization problems come from applications in which there are explicit constraints on the variables. The constraints on the

variables can vary widely from simple bounds to systems of equalities and inequalities that model complex relationships among the variables. Constrained optimization problems can be furthered classified according to the nature of the constraints (e.g., linear, nonlinear, convex) and the smoothness of the functions (e.g., differentiable or non-differentiable). Different kinds of optimization problems usually call for distinct approaches to solve them.

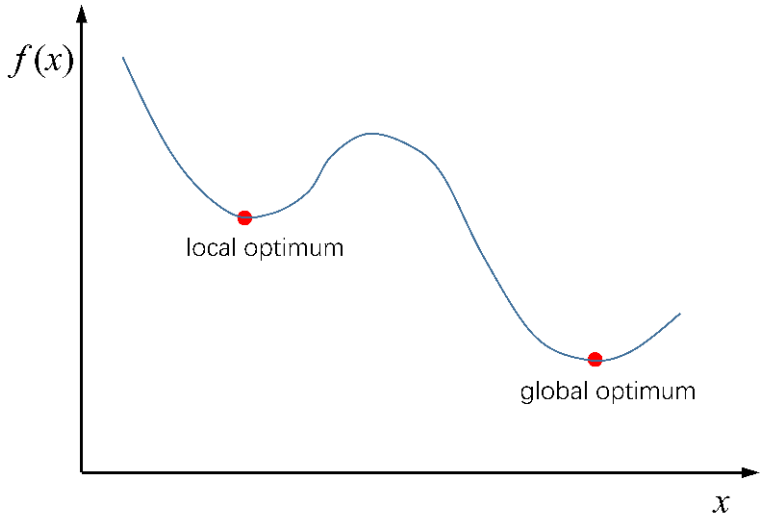

Figure 1.1 Illustration of local and global minimum

In optimization, after an problem is modeling as a mathematical form similar as (1.1), the next essential step is to use the proper algorithm to solve the optimization problem. A brief review of optimization algorithms is given in the following section.

1.2 Overview of Optimization Algorithms

1.2.1 Classification of Optimization Algorithms

Optimization algorithms, which try to find the minimum values of mathematical functions, are essential in optimization. There are many optimization algorithms in the literature and no single algorithm is suitable for all problems. Thus, the right choice of an optimization algorithm is crucially important in solving a given optimization problem. Optimization algorithms can be classified in many ways, depending on the focus or the characteristics we are trying to compare [4]. For instance, optimization algorithms can be

classified as derivative-based and derivative-free, deterministic and stochastic, trajectory-based and population-based, and so on. In this section, the derivative-based and derivative-free algorithms classification is considered. Derivative-based methods (or gradient-based algorithms) use the derivative information in the search process, while the derivative-free methods (gradient-free algorithms) only use use the values of the objective, not any derivatives.

Derivative-based methods generally require a much smaller number of iterations to converge to an optimum compared to derivative-free methods. However, only convergence to local minimum is guaranteed for derivative-based methods, while derivative-free methods are able to find global minimum. In addition, derivative-based methods are limited in cases where the objective function is always differentiable and has continuous derivatives over the search domain. Derivative-free methods do not place limitations on objective functions. Several typical and popular algorithms including both derivative-based and derivative-free algorithms for multi-variable/multi-dimensional unconstrained optimization problems are introduced subsequently.

1.2.2 Derivative-based Algorithms

Since derivative-based algorithms require the first-order and second-order derivatives of the objective function, firstly, some concepts about the derivative are given. The first-order derivative or so-called gradient of a function $f(\mathbf{x})$ at a point $\mathbf{x}=[x_1,\cdots,x_d]^{\mathrm{T}}$ is

$$\nabla f(\mathbf{x})=\left[\frac{\partial f(\mathbf{x})}{\partial x_1},\frac{\partial f(\mathbf{x})}{\partial x_2},\cdots,\frac{\partial f(\mathbf{x})}{\partial x_d}\right]^{\mathrm{T}}. \tag{1.4}$$

The gradient represents the slope of the tangent of the graph of the function. More precisely, the gradient points in the direction of the greatest rate of increase of the function [1]. It is obvious that $f(\mathbf{x})$ decreases fastest in the direction of its negative gradient. Thus, gradient information provides the search direction to locate the minimum of the function.

The Hessian matrix or Hessian $\mathbf{H}(\mathbf{x})$, which represents the second-order partial derivatives of $f(\mathbf{x})$, is written as

$$\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}. \tag{1.5}$$

For unconstrained design space, the conditions for optimal solution can be formulated via the first and the second partial derivatives. The necessary conditions for $\mathbf{x}^*$ to be a minimum of $f(\mathbf{x})$ are $\nabla f(\mathbf{x}^*) = 0$ and the Hessian matrix $\mathbf{H}(\mathbf{x}^*)$ is positive definite [1, 5] ($\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x} > 0$ for any non-zero column vectors $\mathbf{x} \in \mathbb{R}^d$).

In practice, for most optimization problems in which the objective functions $f(\mathbf{x})$ are nonlinear, the derivatives have to be evaluated numerically. Finite difference methods, including forward difference, backward difference, and central difference methods, can be used to calculate the derivative of a function at a point. Because the central difference method for computing the derivative is more accurate than forward/backward difference methods, it is preferable in applications.

1.2.2.1 Steepest Descent Method

The steepest descent or gradient descent algorithms choose search direction as the negative gradient, i.e., $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$ based on the idea that f decreases fastest in the direction of this negative gradient [6]. In successive iterations, the design variables are updated as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)} = \mathbf{x}^{(k)} - \alpha^{(k)}\nabla f(\mathbf{x}^{(k)}), \tag{1.6}$$

where $\alpha^{(k)}$ is called step length, which is a positive scalar parameter that can be determined using the line search algorithm such as the golden section method [1]. The iterative process terminates until the convergence criteria are fulfilled. The algorithm for the steepest descent method is described in Algorithm 1.1.

The steepest descent method ensures a reduction in the objective function value at every iteration. Generally, when the search point is far away from the minimum, the gradient will be higher and the function reduction will be large for an iteration. As approaching the minimum, the gradient value usually decreases, i.e., the method becomes sluggish (slow convergence) near the minimum. To overcome this, the conjugate gradient method was proposed, which selects successive descent directions in a conjugate direction to previous descent directions [7].

Algorithm 1.1 Steepest Descent

1: **input:** objective function $f(\mathbf{x})$, initial point $\mathbf{x}^{(0)}$, tolerance ε, set $k=0$.
2: **repeat**
3: compute the search direction $\mathbf{d}^{(k)} = -\nabla f\left(\mathbf{x}^{(k)}\right)$
4: determine the step length $\alpha^{(k)}$ by minimizing $f\left(\mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}\right)$ using line search
5: update candidate $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$
6: $k \leftarrow k+1$
7: **until** $\left\|\nabla f\left(\mathbf{x}^{(k)}\right)\right\| \le \varepsilon$

1.2.2.2 Newton's Method

Newton's method uses the search direction, which is based on the first-order and second-order derivative information, given by $\mathbf{d}^{(k)} = -\left[\mathbf{H}\left(\mathbf{x}^{(k)}\right)\right]^{-1}\nabla f\left(\mathbf{x}^{(k)}\right)$. Correspondingly, Newton's method updates candidate solutions at each iteration via

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)} = \mathbf{x}^{(k)} - \left[\mathbf{H}\left(\mathbf{x}^{(k)}\right)\right]^{-1}\nabla f\left(\mathbf{x}^{(k)}\right), \tag{1.7}$$

where $\nabla f(\mathbf{x})$ and $\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$ denotes the gradient vector and Hessian matrix of $f(\mathbf{x})$, respectively. The algorithm for Newton's method is described in Algorithm 1.2.

From Equation (1.7), for updating candidate solution at each iteration, the inverse of Hessian matrix need to be computed, which can be expensive. To ease the computational cost, approximations to the Hessian and its inverse are used. This has brought about the so-called quasi-Newton methods, in which the DFP and BFGS methods are popular [6, 8].

Algorithm 1.2 Newton's method

1: **input:** objective function $f(\mathbf{x})$, initial point $\mathbf{x}^{(0)}$, tolerance ε, set $k = 0$.
2: **repeat**
3: compute the search direction $\mathbf{d}^{(k)} = -\left[\mathbf{H}\left(\mathbf{x}^{(k)}\right)\right]^{-1} \nabla f\left(\mathbf{x}^{(k)}\right)$
4: update candidate $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$
5: $k \leftarrow k+1$
6: **until** $\left\|\nabla f\left(\mathbf{x}^{(k)}\right)\right\| \le \varepsilon$

1.2.2.3 Levenberg-Marquardt Method

The advantage of the steepest descent method is that it reaches closer to the minimum of the function in a few iterations even when the starting point is far away from the optimum [1]. However, the method shows sluggishness near the optimum point. On the contrary, Newton's method shows a faster convergence if the starting point is close to the minimum point. Newton's method may not converge if the starting point is far away from the optimum point. These have driven the development of the so-called hybrid methods that take advantages of different methods.

Algorithm 1.3 Levenberg-Marquardt (LM) method

1: **input:** objective function $f(\mathbf{x})$, initial point $\mathbf{x}^{(0)}$, tolerance ε, set $k = 0$.
2: **repeat**
3: compute $f\left(\mathbf{x}^{(k)}\right), \nabla f\left(\mathbf{x}^{(k)}\right)$ and $\mathbf{H}\left(\mathbf{x}^{(k)}\right)$
4: compute the search direction $\mathbf{d}^{(k)} = -\left[\mathbf{H}\left(\mathbf{x}^{(k)}\right) + \lambda \mathbf{I}\right]^{-1} \nabla f\left(\mathbf{x}^{(k)}\right)$
5: update candidate $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$
6: **If** $f\left(\mathbf{x}^{(k+1)}\right) < f\left(\mathbf{x}^{(k)}\right)$ **then**
7: $\lambda \leftarrow \lambda/2$ (change the value of λ)
8: **else**
9: $\lambda \leftarrow 2\lambda$ (change the value of λ)
10: **end if**
11: $k \leftarrow k+1$
12: **until** $\left\|\nabla f\left(\mathbf{x}^{(k)}\right)\right\| \le \varepsilon$

The Levenberg-Marquardt (LM) method is a hybrid method that combines the strength of both the steepest descent and Newton's methods. The search direction in LM method is:

$$\mathbf{d}^{(k)} = -\left[\mathbf{H}\left(\mathbf{x}^{(k)}\right) + \lambda \mathbf{I}\right]^{-1} \nabla f\left(\mathbf{x}^{(k)}\right), \tag{1.8}$$

where $\mathbf{I}$ is an identity matrix and λ is a scalar that is set to a high value at the start of the algorithm. The value of λ is altered during every iteration depending on whether the function value is decreasing or not. If the function value decreases in the iteration, λ decreases by a factor. On the other hand, if the function value is increase in the iteration, λ increases by a factor. The algorithm for the LM method is described in Algorithm 1.3.

1.2.3 Derivative-free Algorithms

Algorithms using derivative information are generally efficient, but these algorithms pose limitations on objective functions. In many real-world optimization problems, the objective function is evaluated by performing computer experiments and there is no analytical expression for the objective function or its derivatives. In these case, derivative-free algorithms are required. Additionally, when discontinuity exists in objective function, derivative-free algorithms may be more efficient and proper. Subsequently, several derivative-free algorithms are presented.

1.2.3.1 Nelder-Mead Simplex Algorithm

The Nelder-Mead algorithm [9] solves the optimization problem by containing the solution within a simplex. Simplex refers to a geometric figure formed by $d+1$ points in a d-dimension space, which is the generalization of a polygon to d-dimension. For simplicity, simplex in the d-dimension space is referred to as d-simplex, which has $d+1$ vertices. The Nelder-Mead algorithm starts with a set of points in $\mathbb{R}^d$ forming a simplex and at each iteration, the objective function is evaluated at the vertice of the simplex. Using this information, the simplex is moved in the search space. The process of moving the simplex is continued until the optimum value of the function is reached. Three basic operations are required to move the simplex in the search space: reflection, contraction, and expansion [1].

Algorithm 1.4 Nelder-Mead Simplex Algorithm

input: objective function $f(\mathbf{x})$, initial simplex $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(d+1)}$, coefficients α, β, γ and δ, tolerance ε.
repeat
  re-order the vertices according to $f\left(\mathbf{x}^{(1)}\right) \le f\left(\mathbf{x}^{(2)}\right) \le \cdots \le f\left(\mathbf{x}^{(d+1)}\right)$
  compute the centroid point $\bar{\mathbf{x}} = \frac{1}{d}\sum_{i=1}^{d} \mathbf{x}^{(i)}$
  generate a trial point through reflection $\mathbf{x}^{(r)} = \bar{\mathbf{x}} + \alpha\left(\bar{\mathbf{x}} - \mathbf{x}^{(d+1)}\right)$
  if $f\left(\mathbf{x}^{(r)}\right) < f\left(\mathbf{x}^{(1)}\right)$ **then**
    compute expansion point $\mathbf{x}^{(e)} = \mathbf{x}^{(r)} + \beta\left(\mathbf{x}^{(r)} - \bar{\mathbf{x}}\right)$
    if $f\left(\mathbf{x}^{(e)}\right) < f\left(\mathbf{x}^{(r)}\right)$ **then**
      accepted the expansion point $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(e)}$
    else
      accept the reflection point $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(r)}$
    end if
  else if $f\left(\mathbf{x}^{(r)}\right) < f\left(\mathbf{x}^{(d)}\right)$ **then**
    accept reflection point $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(r)}$
  else if $f\left(\mathbf{x}^{(r)}\right) \ge f\left(\mathbf{x}^{(d)}\right)$ **then**
    contraction $\mathbf{x}^{(c)} = \mathbf{x}^{(d+1)} + \gamma\left(\bar{\mathbf{x}} - \mathbf{x}^{(d+1)}\right)$
    if $f\left(\mathbf{x}^{(c)}\right) < f\left(\mathbf{x}^{(d+1)}\right)$ **then**
      accept the contraction point $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(c)}$
    else
      reduction (shrink) $\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(1)} + \delta\left(\mathbf{x}^{(i)} - \mathbf{x}^{(1)}\right),\ i = 2, 3, \cdots, d$
    end if
  end if
until $\sqrt{\frac{1}{d+1}\sum_{i=1}^{d+1}\left[f\left(\mathbf{x}^{(i)}\right) - f\left(\bar{\mathbf{x}}\right)\right]^2} \le \varepsilon$

The fundamental procedure of Nelder-Mead algorithm is described in the following. The first step is to construct an initial *d*-simplex with $d+1$ vertices and to evaluate the objective function at the vertices. Then, the $d+1$ vertices are ordered according to their corresponding objective function values as

$$f\left(\mathbf{x}^{(1)}\right) \le f\left(\mathbf{x}^{(2)}\right) \le \cdots \le f\left(\mathbf{x}^{(d+1)}\right), \tag{1.9}$$

where $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(d+1)}$ are ordered vertices of the simplex, whose objective function values are in ascending order. Apparently, $\mathbf{x}^{(d+1)}$ is the worse point (solution) and $\mathbf{x}^{(1)}$ is the best solution, which is the convention in simplex method for minimization problem. At each iteration, similar ranking manipulations are performed.

Then, the centroid point $\bar{\mathbf{x}}$ of simplex is computed using all the vertices but with the exclusion of the worst vertex $\mathbf{x}^{(d+1)}$. That is

$$\bar{\mathbf{x}} = \frac{1}{d}\sum_{i=1}^{d} \mathbf{x}^{(i)}. \tag{1.10}$$

Using the centroid point as the basis point, the reflection point of the worse point $\mathbf{x}^{(d+1)}$ is computed as

$$\mathbf{x}^{(r)} = \bar{\mathbf{x}} + \alpha\left(\bar{\mathbf{x}} - \mathbf{x}^{(d+1)}\right), \quad \alpha > 0, \tag{1.11}$$

where α is the so-called reflection coefficient. The typically value of $\alpha = 1$ is often used.

Whether the new trial solution is accepted or not and how to update the new vertex, depends on the objective function value at $\mathbf{x}^{(r)}$. There are three possibilities [10]:

- If $f\left(\mathbf{x}^{(1)}\right) \le f\left(\mathbf{x}^{(r)}\right) < f\left(\mathbf{x}^{(d)}\right)$, then replace the worst vertex $\mathbf{x}^{(d+1)}$ by $\mathbf{x}^{(r)}$, i.e., $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(r)}$.
- If $f\left(\mathbf{x}^{(r)}\right) < f\left(\mathbf{x}^{(1)}\right)$ which means the objective improves, then it is possible to move the vertex further along the line of reflection to seek an expand point that can improve the objective even further. The expansion point is computed as

  $$\mathbf{x}^{(e)} = \mathbf{x}^{(r)} + \beta\left(\mathbf{x}^{(r)} - \bar{\mathbf{x}}\right), \tag{1.12}$$

  where β is the expansion coefficient. Typically, $\beta = 2$ is adopted. Now, we have to check if $f\left(\mathbf{x}^{(e)}\right)$ improves even better. If $f\left(\mathbf{x}^{(e)}\right) < f\left(\mathbf{x}^{(r)}\right)$, it is accepted and the vertex is updated as $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(e)}$; otherwise, the result of reflection is used, i.e., $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(r)}$.
- If the reflect point is not better than the second worst point, i.e., $f\left(\mathbf{x}^{(r)}\right) \ge f\left(\mathbf{x}^{(d)}\right)$, the contraction operation, which reduce the size of the simplex while maintaining the best sides, is performed. The contraction point is computed as

$$\mathbf{x}^{(c)} = \mathbf{x}^{(d+1)} + \gamma\left(\bar{\mathbf{x}} - \mathbf{x}^{(d+1)}\right), \quad 0 < \gamma < 1, \tag{1.13}$$

where γ is the so-called contraction coefficient and $\gamma = 1/2$ is usually used. If $f\left(\mathbf{x}^{(c)}\right) < f\left(\mathbf{x}^{(d+1)}\right)$ is true, we then update $\mathbf{x}^{(d+1)} \leftarrow \mathbf{x}^{(c)}$.

If all the above steps fail, we should reduce (shrink) the size of the simplex towards the best vertice $\mathbf{x}^{(1)}$. This is called the reduction (or shrink) step, which is expressed by

$$\mathbf{x}^{(i)} \leftarrow \mathbf{x}^{(1)} + \delta\left(\mathbf{x}^{(i)} - \mathbf{x}^{(1)}\right), \ i = 2, 3, \cdots, d, \tag{1.14}$$

where δ is called the shrink coefficient, and usually $\delta = 1/2$ is used. The preceding operations are continued until the the standard deviation of the objective function values computed at the vertices of the simplex becomes less than the tolerance ε, i.e.,

$$\sqrt{\frac{1}{d+1}\sum_{i=1}^{d+1}\left[f\left(\mathbf{x}^{(i)}\right) - f\left(\bar{\mathbf{x}}\right)\right]^2} \le \varepsilon. \tag{1.15}$$

Above described Nelder-Mead simplex algorithm is described in Algorithm 1.4.

1.2.3.2 Trust-Region Method

The so-called trust-region method is one of the most widely used optimization algorithms. This method uses a model, which is usually smooth, easy to evaluate, and presumed to be accurate in a neighborhood (trust-region) about the current iteration, to approximate the objective function [11]. The next trial solution is then found by using the approximate model of objective function. Since the approximate model of objective function is smooth and easy to evaluate, finding the next solution based on approximation is cheaper than that from original function.

The fundamental step in trust-region algorithm is to approximate the objective function. The commonly used approximate model is the quadratic model [12], which is of the form:

$$\phi^{(k)}(\mathbf{x}) = f\left(\mathbf{x}^{(k)}\right) + \left[\mathbf{g}^{(k)}\right]^{\mathrm{T}}\left(\mathbf{x} - \mathbf{x}^{(k)}\right) + \frac{1}{2}\left(\mathbf{x} - \mathbf{x}^{(k)}\right)^{\mathrm{T}}\mathbf{H}^{(k)}\left(\mathbf{x} - \mathbf{x}^{(k)}\right), \tag{1.16}$$

and this approximation is valid only in a small neighborhood of $\mathbf{x}^{(k)}$, which is the so-called trust-region, defined by

$$\Omega^{(k)} = \left\{\mathbf{x} \in \mathbb{R}^d \,\middle|\, \left\|\mathbf{x} - \mathbf{x}^{(k)}\right\| \le \delta^{(k)}\right\}, \tag{1.17}$$

where $\mathbf{x}^{(k)}$ is the solution of current iterate, $\mathbf{g}^{(k)} \in \mathbb{R}^d$ and symmetric matrix $\mathbf{H}^{(k)} \in \mathbb{R}^{d \times d}$ are parameters for the approximate model $\phi^{(k)}(\mathbf{x})$, and $\delta^{(k)}$ is the trust-region radius. Different methods can be used to determine $\mathbf{g}^{(k)}$ and $\mathbf{H}^{(k)}$. Usually, $\mathbf{g}^{(k)}$ and $\mathbf{H}^{(k)}$ are determined by the first and second order derivatives of $f(\mathbf{x})$. However, considering trust-region algorithm as a derivative-free method, $\mathbf{g}^{(k)}$ and $\mathbf{H}^{(k)}$ can be estimated by approximate derivatives (e.g., through finite difference methods) or by requiring $\phi^{(k)}(\mathbf{x})$ to interpolate a set of sample points as in [12].

Let $\mathbf{x}^{(*)}$ denotes the minimum of approximation model $\phi^{(k)}$ in the trust-region $\Omega^{(k)}$, that is

$$\begin{aligned} \mathbf{x}^{(*)} &= \arg\min_{\mathbf{x} \in \Omega^{(k)}} \phi^{(k)}(\mathbf{x}) \\ &= f\left(\mathbf{x}^{(k)}\right) + \left[\mathbf{g}^{(k)}\right]^{\mathrm{T}} \left(\mathbf{x} - \mathbf{x}^{(k)}\right) + \frac{1}{2}\left(\mathbf{x} - \mathbf{x}^{(k)}\right)^{\mathrm{T}} \mathbf{H}^{(k)} \left(\mathbf{x} - \mathbf{x}^{(k)}\right). \end{aligned} \tag{1.18}$$

How good the approximate model $\phi^{(k)}(\mathbf{x})$ is to the actual objective function $f(\mathbf{x})$ in the trust-region can be measured by the ratio between the actual reduction and the predicted reduction

$$r^{(k)} = \frac{f\left(\mathbf{x}^{(k)}\right) - f\left(\mathbf{x}^{(*)}\right)}{\phi^{(k)}\left(\mathbf{x}^{(k)}\right) - \phi^{(k)}\left(\mathbf{x}^{(*)}\right)}. \tag{1.19}$$

Now, the two key questions are how to update the new solution $\mathbf{x}^{(k+1)}$, which is the center of the newly updated trust region, and the radius $\delta^{(k+1)}$ of the trust region for next iteration. The scheme for these updating are as following:

- If $0 < \eta_1 \le r^{(k)} \le \eta_2 < 1$, which means the approximate model is appropriate, we accept $\mathbf{x}^{(*)}$ as the next solution, i.e., $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(*)}$, and keep the size of the trust region, that is $\delta^{(k+1)} \leftarrow \delta^{(k)}$.
- If $r^{(k)} < \eta_1$, which implies that the approximate model is not appropriate in the region, thus, we do not accept the solution $\mathbf{x}^{(*)}$ and the radius of the trust region is reduced by a factor $\gamma_1 < 1$. That is, $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)}$, $\delta^{(k+1)} \leftarrow \gamma_1 \delta^{(k)}$.

- If $r^{(k)} > \eta_2$, which denotes that the approximate model is very appropriate, correspondingly, we do accept the solution $\mathbf{x}^{(*)}$ and enlarge the trust region by a factor $\gamma_2 > 1$, i.e., $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(*)}$ and $\delta^{(k+1)} \leftarrow \gamma_2 \delta^{(k)}$.

Typical values for constants in the algorithm are $\eta_1 = 0.25$, $\eta_2 = 0.75$, $\gamma_1 = 0.5$ and $\gamma_2 = 3$. According to above described procedure, the trust-region moves and updates iteratively until the optimality is found or a fixed number of iterations is reached, which is presented in Algorithm 1.5.

Algorithm 1.5 Trust-Region Algorithm

input: objective function $f(\mathbf{x})$, initial point $\mathbf{x}^{(0)}$, initial trust-region radius $\delta^{(0)}$, initial algorithm constants $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2$, and tolerance ε; set $k = 0$.
repeat
  construct an approximate model $\phi^{(k)}(\mathbf{x})$ in Equation (1.16) for the objective function $f(\mathbf{x})$ in current trust-region $\Omega^{(k)} = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \left\| \mathbf{x} - \mathbf{x}^{(k)} \right\| \le \delta^{(k)} \right\}$.
  find the minimum of $\phi^{(k)}(\mathbf{x})$ in $\Omega^{(k)}$, $\mathbf{x}^{(*)} = \arg\min_{\mathbf{x} \in \Omega^{(k)}} \phi^{(k)}(\mathbf{x})$.
  compute the ratio $r^{(k)} = \left[f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(*)}) \right] \Big/ \left[\phi^{(k)}(\mathbf{x}^{(k)}) - \phi^{(k)}(\mathbf{x}^{(*)}) \right]$.
  if $r^{(k)} < \eta_1$ **then**
    reject the step and reduce the trust-region: $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)}$, $\delta^{(k+1)} \leftarrow \gamma_1 \delta^{(k)}$
  else if $r^{(k)} \le \eta_2$ **then**
    accept the step and keep the trust-region: $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(*)}$, $\delta^{(k+1)} \leftarrow \delta^{(k)}$
  else if $r^{(k)} > \eta_2$ **then**
    accept the step and enlarge the trust-region : $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(*)}$, $\delta^{(k+1)} \leftarrow \gamma_2 \delta^{(k)}$
  end if
  $k \leftarrow k + 1$
until $\left\| \nabla f(\mathbf{x}^{(k)}) \right\| \le \varepsilon$

1.2.3.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of stochastic derivative-free optimization methods. EAs use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Among EAs, genetic algorithms (GAs), evolution strategies (ESs) and differential evolution (DE) are popular and commonly applied in research and engineering problems.

EAs use population-based search mechanism. The general framework of an evolutionary algorithm is presented in Algorithm 1.6. In the initialization of an EA, the first generation (parent population), consisting of one or more individuals, is created, and the fitness values (the objective function value is called fitness in EAs) of its individuals are evaluated. Then, the so-called evolution loop is entered, which consists of the evolution operators recombination (also known as crossover), mutation, evaluation, and selection [3]. Recombination creates new individuals, also called offspring, from the parent population. Mutation adds random changes to the newly created offspring. Subsequently, the fitness of the offspring is evaluated. Based on the fitness, selection identifies a subset of individuals which form the new parent population for the next iteration of the evolution loop. The loop is stopped when the termination criterion is fulfilled.

Algorithm 1.6 General Framework of an Evolutionary Algorithm

```
initialization
repeat
    Recombination
    Mutation
    Evaluation
    Selection
until termination criterion fulfilled
```

1.2.3.4 Other Derivative-free Methods

In addition to the Nelder-Mead method, Trust-region method and EAs that we introduced above, there are numerous other derivative-free optimization methods. As deterministic methods, there are direction search methods including Generalized pattern search method [13], Mesh adaptive direct search method [14], Lipschitzian-based methods, such as DIRECT algorithm [15] and Brand-and-bound search, and multilevel coordinate search [16], etc. For methods that mimic natural processes or some other physical analogies, besides EAs, there are simulated annealing (SA) algorithm, particle swarm optimization (PSO) algorithm, ant colony optimization (ACO) algorithm, harmony search (HS) algorithm, and so forth.

1.3 Review of Surrogate-Assisted Evolutionary Optimization

Surrogate-assisted evolutionary algorithms are mainly motivated from reducing computational cost in evolutionary optimization of expensive problems. In surrogate-assisted evolutionary algorithms, the so-called surrogate models or metamodels are constructed to simulate the behavior of the original (true) fitness function. Because surrogate models are much cheaper to evaluate, the computational costs are reduced with the assist of surrogate models. In recent years, surrogate-assisted evolution methods based on fitness approximation are preferable and popular in real-world applications, especially in expensive optimization problems. Mainly based on the review papers [17–19] and book [2], this section briefly reviews surrogate-assisted evolutionary optimization based on fitness approximation.

Three main aspects of surrogate-assisted evolutionary optimization are: types of fitness approximation methods, the working styles and the management schemes of the fitness approximation [20]. The following of this section expands these three aspects successively.

1.3.1 Fitness Approximation Methods

The popular and most commonly used fitness approximation methods are using so-called surrogate models or metamodels that are constructed based on a set of evaluated points from the evaluation history. Based on machine learning and statistical learning techniques, so far, several models have been used for fitness approximation. The most popular ones including polynomial regression, Kriging model, radial basis functions, neural networks and support vector machines. A comprehensive description of surrogate modeling techniques are presented in Chapter 3.

1.3.2 Working Styles of Fitness Approximation

The working style of fitness approximation denotes the mechanism of incorporating the fitness approximation models (surrogate models) into evolutionary algorithms (EAs). Surrogate models can be embedded in almost every operations of evolutionary algorithms, such as initialization, mutation, recombination and fitness evaluations [18]. According to

[20], the incorporation mechanisms of surrogate in EAs can be divided into direct and indirect fitness replacement methods, i.e., direct and indirect styles, as shown in Figure 1.2. The direct fitness replacement method is to use the approximate fitness to directly replace the original (exact) fitness during the evolutionary optimization. Individuals mostly have the approximate fitness during the optimization. Only a few individuals are evaluated by original fitness function for control purpose. The indirect fitness replacement method is to use the approximate fitness only for some but not all processes in the EAs, such as population initialization and EA operators. The original fitness is kept for each individual and the approximate fitness is not used to directly replace the original fitness.

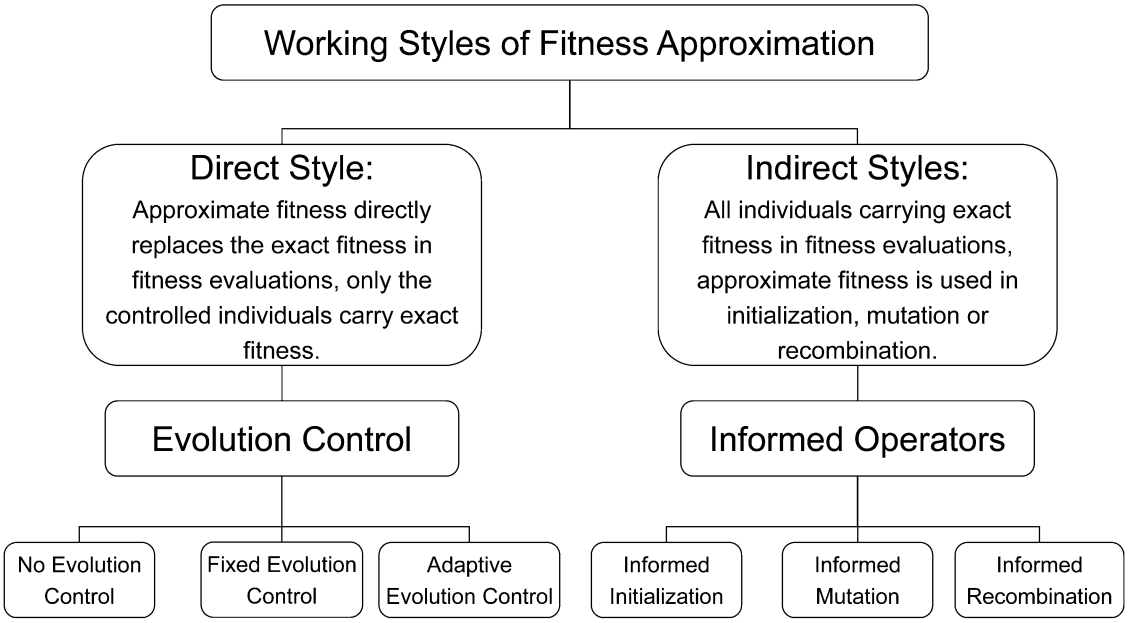

Figure 1.2 Working styles of fitness approximation

1.3.2.1 Direct Style

Direct style, i.e., direct fitness replacement method, is a straightforward strategy to use fitness approximation models. Individuals are evaluated by surrogate models and then the estimated fitness is assigned to each individual. In other words, the fitness approximation model (surrogate model) undertakes the role of the original fitness function, and thus the original fitness is replaced by approximate fitness. However, the obvious drawback is that the evolutionary algorithm may be misled to false optimum due to the inaccuracy of the

surrogates of the original fitness function [21]. A false optimum is an optimum of the surrogate model, which is not an optimum of the original fitness function [18]. Therefore, in most cases, surrogates should be used together with the original fitness function in order to prevent the optimization process from being misled by false optima introduced by surrogates. This is termed as model management [22, 23] or evolution control [21, 24], which is of significant importance in direct fitness replacement methods. The details of model management will be described below in Section 1.3.3.

1.3.2.2 Indirect Style

In indirect style, i.e., indirect fitness replacement method, the exact fitness of each individual is computed during EA process and the approximate fitness is used in other ways. For example, the approximate fitness can be used for population pre-selection. In [25], the Gaussian process model was used to pre-select the most promising solutions, which were then actually evaluated by the original fitness function. In this method, based on a standard (μ, λ)-ES , $\lambda' > \lambda$ new offspring individuals are firstly created from μ parents. Then, λ individuals out of these λ' individuals are pre-selected according the merit function values evaluated by Gaussian process model to generate the offspring population. The pre-selected λ offspring are evaluated by original fitness function for selection and recombination.

According to [20, 26, 27], indirect fitness replacement method uses approximate fitness in mutation and recombination (crossover) operators through a technique known as Informed Operators. In this approach, the approximate models are used to evaluated the fitness of candidates only during the mutation and/or recombination process. After the mutation and/or mutation process, the exact fitness is still computed for each individuals. The advantage is that using the approximate fitness indirectly in the form of informed operator rather than direct fitness replacement is expected to keep the optimization moving toward the true global optima and to reduce the risk of convergence to suboptimal solutions or false optima because each individual in the population still carries its exact fitness [27]. Some of the informed operators in [26, 27] are described as follows:

- **Informed initialization:** An individual in the initial population is generated by selecting the best individual from a number of uniformly distributed individuals in the design space according to the approximate fitness.
- **Informed mutation:** The informed mutation is best mutation from several random mutations according to the approximate fitness. Specifically, several random mutations of the base point are firstly generated ; these mutations then are evaluated by the surrogate model; the mutation with the best approximate fitness is returned as the result.
- **Informed recombination:** Two parents are selected randomly according to the usual selection strategy. These two parents are not changed in the course of the informed recombination (crossover) operation. Several recombinations are conducted by randomly selecting a crossover method, randomly selecting its internal parameters and applying it to the two parents to generate a potential child. The surrogate is used to evaluate every potential child, and the best child is selected as the outcome.

1.3.3 Model Management

The direct working style of fitness approximation uses surrogate model for fitness evaluations and may reduce the number of fitness evaluations significantly [18]. However, the application of surrogate models to evolution computation is not as straightforward as one may expect. Apparently, there are two principles in direct fitness replacement strategy. First, it should be ensured that the evolutionary algorithm converges to the global optimum or a near-optimum of the original fitness function. Second, the computational cost should be reduced as much as possible. In other words, the number of evaluating the expensive original fitness function should be decreased as much as possible.

Due to the lack of data and the high dimensionality of design space, it is very difficult to construct a perfect global approximate model of the original fitness function. The inaccuracy of the approximate model may lead the EA to inferior local optimum or false

optimum. To tackle this problem, two main aspects are considered. Firstly, the approximate model should be used together with the original fitness function. This is known as evolution control in evolutionary computation [21]. Secondly, the quality of surrogate model should be improved as much as possible with the given limited data. Several aspects are important to improve the model quality, such as selection of the model, selection of surrogate model training method and selection of error measures [17]. Therefore, model management, which consists of above two aspects, is essential for direct fitness replacement methods.

1.3.3.1 Evolution Control

Evolution control means that, in surrogate-assisted evolutionary computation, the original fitness function is used to evaluate some/all individuals in some/all generations [21]. An individual that is evaluated using original fitness function is called a controlled individual. Similarly, a generation in which all its individuals are evaluated using the original fitness function is called a controlled generation [17]. The evolution control in surrogate-assisted evolutionary computation generally can be divided into three main approaches [17, 18].

No Evolution Control

When the built surrogate model is assumed of high-fidelity, the original fitness function is not used in evolutionary computation, i.e., no individual or generation is controlled. This is known as no evolution control. In this case, after the surrogate model is constructed, the original fitness function is not at all used in evolutionary computation.

Fixed Evolution Control

Fixed evolution control implies that the frequency of evolution control is fixed. There are two approaches to fixed evolution control: individual-based evolution control and generation-based evolution control. Individual-based evolution control means that in each generation, some of the individuals are evaluated by surrogate model and the others are evaluated using the original fitness function, which is illustrated in Figure 1.3. In individual-based evolution control, the individual selection can be random or using some strategy, e.g., selecting the best individual (based on the prediction of surrogate model) for evolution

control. In generation-based evolution control, all individuals in a selected generation will be evaluated by the original fitness function. The generation selection can be random or with a fixed frequency. A generation-based evolution control is shown in Figure 1.4.

The main drawback of fixed evolution control methods is that the frequency of evolution control is fixed. This is not very practical because the fidelity of the surrogate model may vary significantly during optimization process. As a matter of fact, a predefined fixed evolution control frequency may cause strong oscillation during optimization due to large model error [28].

Adaptive Evolution Control

It is intuitive and reasonable to assume that the frequency of evolution control should depend on the fidelity of the surrogate model. This strategy is called adaptive evolution control. In adaptive evolution control, individuals are iteratively controlled to update the surrogate model until the fidelity (or quality) of the surrogate model is acceptable.

1.3.3.2 Off-line Model Training

Off-line model training denotes the training process before the model is used in evolutionary computation [17]. Off-line model training constructs surrogate models based on data sampling (design of experiments) or previous optimization history data. In this case, either the surrogate model is of high fidelity or the original fitness function is expensive to evaluate, so that the original fitness is not used after the model training [20].

1.3.3.3 On-line Model Updating

On-line model training (updating) denotes rebuilding or updating the model during the evolutionary process. In surrogate-assisted evolutionary computation, surrogate model may be constructed at an early stage of the EA process. Because the set of sample points, which are used to train the model, is limited and does not cover the whole search space, the surrogate may concentrate on the region spanned by the existing sample points and not cover the rest of the search space well. As the EA continues and new individuals enter into the population, the accuracy of the previously built surrogate model will decrease. By evolution

control, new points are added into the training data set. Thus the surrogate model needs to be retrained using the old sample points together with the new sample points. This is the typical on-line model updating technique in surrogate-assisted evolutionary computation.

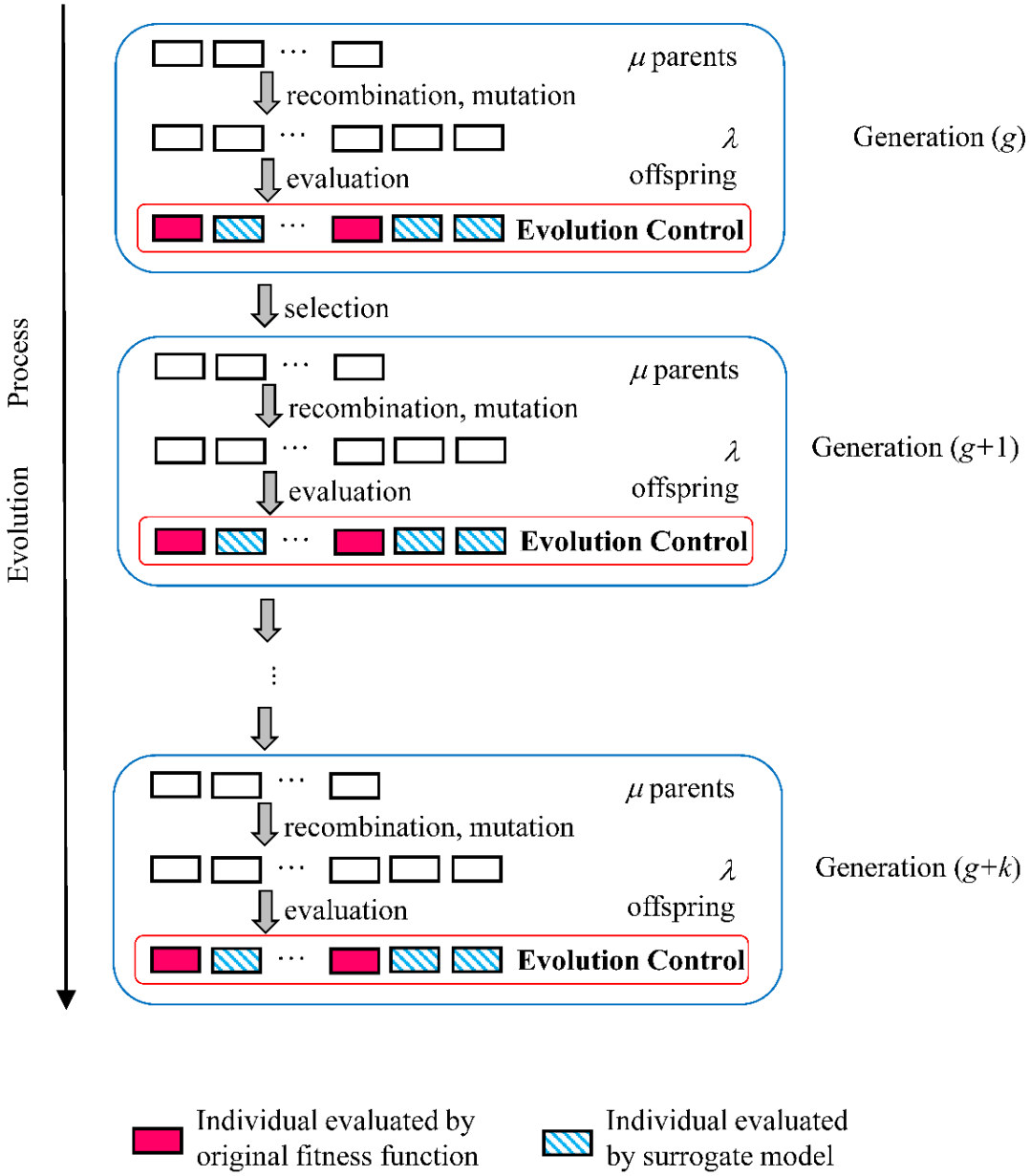

Figure 1.3 Illustration of fixed individual-based control

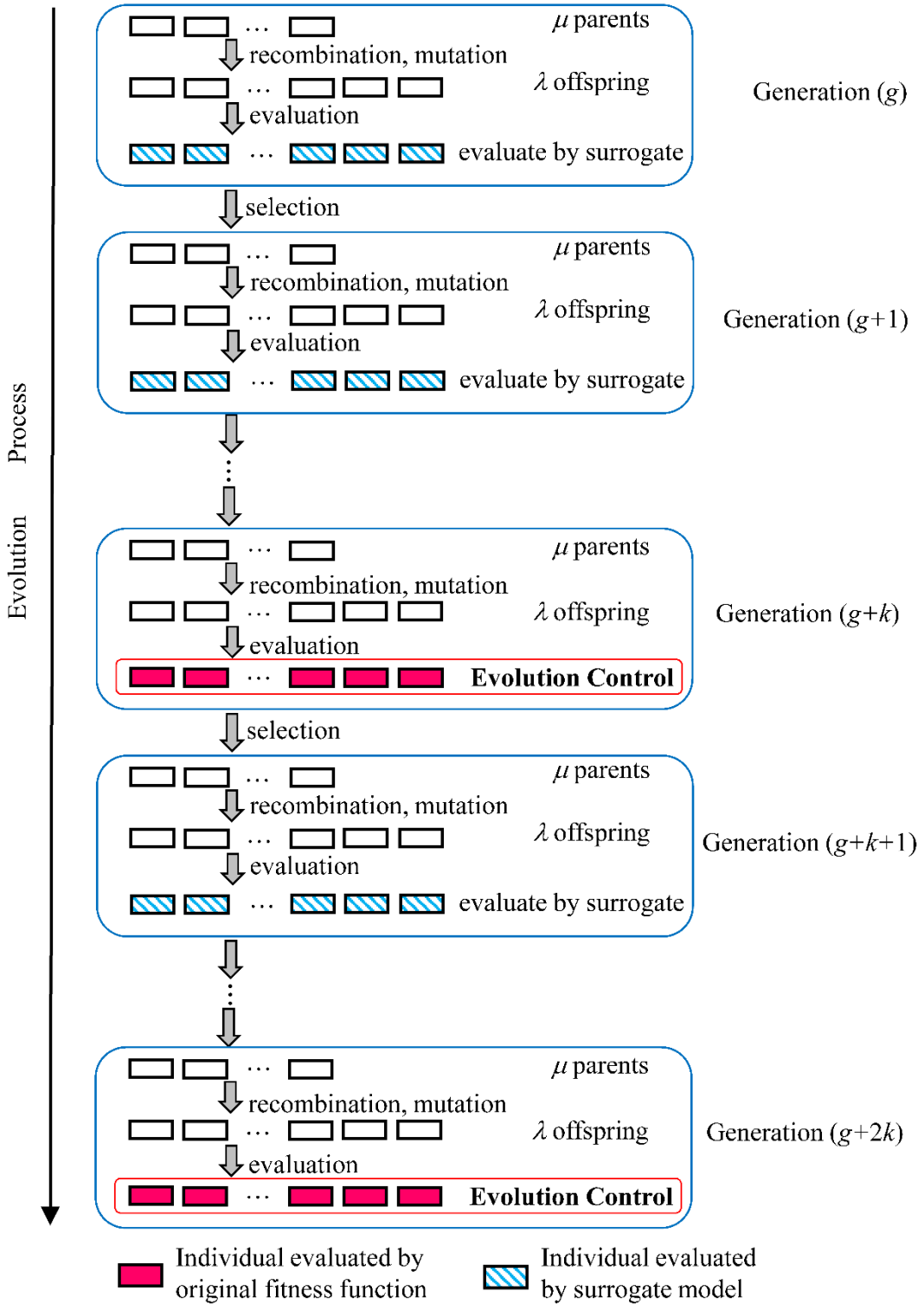

Figure 1.4 Illustration of fixed generation-based control

2. Evolution Strategies

This chapter gives a comprehensive description of evolution strategies (ESs). A short introduction of ESs is provided in Section 2.1. Then the main principles and evolutionary operators used in ESs are introduced in Section 2.2. Subsequently, Section 2.3 presents parameter control of ESs with three ES algorithms, (1+1)-ES, $(\mu/\mu_I, \lambda)$-ES with Cumulative Step-Size Adaptation, and Covariance Matrix Adaptation Evolution Strategy (CMA-ES).

2.1 Introduction

Evolution strategies (ESs) are stochastic, derivative-free optimization methods. It has been proven that ESs are appropriate and successful for continuous black-box optimization, i.e., for optimization scenarios, where no analytical expressions of the objective functions are explicitly given and derivatives are unavailable [29]. Additionally, ESs have the capability of accessing the global optimum for multimodal problems [30]. Therefore, ESs are increasingly popular in solving real-world optimization problems.

ESs are search paradigms inspired by the principles of biological evolution. They belong to the family of evolutionary algorithms (EAs) that address optimization problems by implementing a repeated process of stochastic variations followed by selection. In each generation (iteration), new offspring (candidate solutions) are generated from their parents (candidate solutions that have already been visited), their fitness are evaluated, and then the better offspring are selected to become the parents for the next generation [31].

ESs most commonly address the problem of continuous black-box optimization. The search space is the continuous domain, $\mathbb{R}^d$, and solutions in search space are *d*-dimensional vectors, denoted as $\mathbf{x} \in \mathbb{R}^d$. The objective or fitness function $f : \mathbb{R}^d \to \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$ is considered to be minimized. In the field of evolutionary algorithms, the objective function

is often called fitness function, correspondingly, the objective function value is called the fitness value. High fitness means low fitness function values in the convention minimizing problem. There is no specific assumptions on fitness function f, except that the fitness function f can be evaluated for each $\mathbf{x}$. This kind of search problem is referred to as black-box optimization. The objective is to generate solutions ($\mathbf{x}$ vector) with small fitness values while using a small number of fitness function evaluations.

2.2 Main Principles and Evolutionary Operators

Inspired by the principles of biological evolution, evolution strategies use a repeated process that consist of evolutionary operators which mimics the mechanisms of the Darwinian theory of evolution to address optimization problems. A population $\mathcal{P}$ consists of the so-called individuals is undergoes evolution cycles during each generation (iteration) of evolutionary search. Each individual consists of a candidate solution or vectors of input parameters $\mathbf{x} \in \mathbb{R}^d$, an associated fitness value $f(\mathbf{x})$, and of the so-called endogenous parameters, which are strategy parameters for the mutation operator [3]. In some cases, the population contains only one individuals. Individuals are also denoted as parents or offspring, depending on the context. Generally, a generation loop can be described as :

1. One or several parents are selected from the population (mating selection) and new offspring are generated by recombination of these parents.
2. The new offspring undergo mutation and become new members of the population.
3. Environmental selection reduces the population to its original size by choosing better individuals.

The sequence of generations is continued until a termination criterion is met. Typical termination criteria are set as reaching a maximum number of evaluations, reaching a target fitness value, or stagnation of the search process.

The main principles within above generational procedure of evolution strategies are firstly presented below. Then the corresponding evolutionary operators, which are implementations of main principles in algorithms, used in evolution strategies are described.

2.2.1 Main Principles

2.2.1.1 Environmental Selection

In evolution strategies, environmental selection is applied as so-called truncation selection. Specifically, based on the individuals' fitness $f(\mathbf{x})$, only μ best individuals from the population survive. In other words, only μ individuals that have best fitness are selected and kept in the population. In contrast to roulette wheel selection in genetic algorithms [32], only fitness ranks are used in truncation selection. In evolution strategies, environmental selection is deterministic. Environmental selection, on average, increases the fitness of the population and at the same time reduces diversity as some individuals are discarded. Overage individuals can also removed by environmental selection.

2.2.1.2 Mating Selection and Recombination

In biological filed, recombination, also known as crossover, mixes the genetic material of parents. Similarly, in evolution strategies, recombination combines information from several parents to generate a new offspring. Firstly, mating selection picks individuals from the population to become new parents. These parents are then used to generates a single new offspring by recombination. There are two common scenarios for mating selection and recombination [31]:

- **Fitness-independent** mating selection and recombination do not depend on the fitness values of the individuals and can be either deterministic or stochastic. In this scenario, environmental selection is essential to drive the evolution toward better solutions.
- **Fitness-based** mating selection and recombination, where the recombination operator utilizes the fitness ranking of the parents. Thus, recombination performs in

a deterministic way. Environmental selection can be potentially be omitted in this case.

2.2.1.3 Mutation and Parameter Control

Mutation introduces small, random, and unbiased changes (perturbations) to an individual. This provides the main source of variation (evolutionary changes) of individuals in evolution strategies. Thus, mutation plays an important role in evolution strategies. The average size of these changes introduced by mutation, depends on endogenous parameters that change over time, for example, the step-size σ and covariance matrix $\mathbf{C}$. These parameters are also called control parameters, or endogenous strategy parameters. In contrast, exogenous strategy parameters are fixed once and for all, for example, the parent number μ . Parameter control, i.e., how to control the endogenous parameters, is not always directly inspired by biological evolution, but it is an indispensable and central feature of evolution strategies.

2.2.2 Evolutionary Operators in Evolution Strategies

In order to realize algorithms of evolution strategies, principles described in above subsection need to be specified in concrete evolutionary operators so that to form evolutionary search schemes. The selection, recombination and mutation operators used in evolution strategies are explained below.

2.2.2.1 Selection Operators

Selection operator, which relates directly to the environmental selection concept of survival of the fittest, gives the evolutionary search a direction. In each generation, the selection operator emphasizes better solutions by selecting a subset of individuals based on their fitness to form the new parent population used in the next generation. Usually, we use μ to denotes the number of individuals in parent population (also called the size of parent population), and λ to denotes the number of offspring generated in each generation. In

evolution strategies, the commonly used selection strategies are the comma (indicated by ,) selection and plus (indicated by +) selection:

- **Plus selection** selects the μ-best solutions from the union of the last parent population (μ individuals in parent population) and the current offspring population (λ individuals in offspring population), and is correspondingly denoted as $(\mu+\lambda)$-ES.
- **Comma selection**, i.e., (μ,λ)-ES, selects μ-best solutions exclusively from the offspring population (λ individuals in offspring population), neglecting the parent population, even if the parents have a superior fitness.

Neglecting superior solutions, in comma selection, may sound irrational. However, good solutions can be only local optimum. By using comma selection, the evolutionary process could escape from the local optimum and reach a better optimum [29]. In contrast, when using plus selection, the search process may fail to leave the local optimum without the ability to neglect [30]. Thus, comma selection is advantageous in the case of multimodal problems.

2.2.2.2 Recombination Operators

In evolution strategies, recombination combines information from several parents to generate a new offspring. Usually, multi-parent recombination is used, where more than two parents are combined. Here we use ρ to denotes the number of parent individuals used in recombination (i.e., ρ out of μ parent individuals are used to generate a new offspring by recombination, where $\rho \leq \mu$). The most important and commonly used types of recombination in evolution strategies include:

- **Dominant recombination**, denoted by $(\mu/\rho_D,\lambda)$, is also known as discrete recombination. In dominant recombination, a property of a parent individual is inherited by the offspring, i.e., this property dominates the corresponding property of other parent individuals. Specifically, for each variable (component of the design variable vector $\mathbf{x}$), a single parent is drawn randomly from all ρ parents

to inherit the variable value. With ρ parents $\mathbf{x}_1, \cdots, \mathbf{x}_\rho$, dominant recombination creates the offspring $\mathbf{x}' = \left[x_1', \cdots, x_d'\right]^{\mathrm{T}}$ by randomly choosing the *j*-th component

$$x_j' = x_{ij}, \quad j \in \text{random}\{1, \cdots, \rho\}, \tag{2.1}$$

where x_{ij} is the *j*-th component of the *i*-th parent individual.

- **Intermediate recombination**, denoted by $(\mu/\rho_I, \lambda)$, takes the average value of all the parents. Given ρ parents $\mathbf{x}_1, \cdots, \mathbf{x}_\rho$, each component of the offspring $\mathbf{x}' = \left[x_1', \cdots, x_d'\right]^{\mathrm{T}}$ is the arithmetic mean of the corresponding components of all ρ parents, i.e.,

$$x_j' = \frac{1}{\rho}\sum_{i=1}^{\rho} x_{ij}, \text{ or } \mathbf{x}' = \frac{1}{\rho}\sum_{i=1}^{\rho} \mathbf{x}_i. \tag{2.2}$$

  In this case, the offspring is the centroid of ρ parents.

- **Weighted recombination**, denoted by $(\mu/\rho_W, \lambda)$, is the generalization of intermediate recombination, usually with $\rho = \mu$ in ES. Weighted recombination takes a weighted average of all ρ parents $\mathbf{x}_1, \cdots, \mathbf{x}_\rho$ as the offspring $\mathbf{x}' = \left[x_1', \cdots, x_d'\right]^{\mathrm{T}}$, i.e.,

$$\mathbf{x}' = \sum_{i=1}^{\rho} w_i \mathbf{x}_i, \tag{2.3}$$

  where $w_i, i = 1, \cdots, \rho$ are the weights, which usually have $\sum_{i=1}^{\rho} w_i = 1$. In ES, the weight values depend on the fitness ranking such that parents with higher fitness never get smaller weights than lower ones. With equal weights, weighted recombination recovers to intermediate recombination.

In evolution strategies, the intermediate or weighted recombination with $\rho = \mu$ is often used. Thus, the result of selection and recombination is often deterministic. This means that eventually all offspring are generated by mutation from the same single solution vector (the parent centroid, generated by recombination).

2.2.2.3 Mutation Operators

The mutation operator, which introduces variations by adding perturbation to the result of recombination, provides the main source of variation (evolutionary changes) of offspring in evolution strategies. According to [33, 34], a mutation operator is supposed to fulfill three rules, namely:

- **Reachability:** from an arbitrary solution, any point in the search space can be reachable with probability strictly larger than zero by means of a finite number of applications of the mutation operator.
- **Unbiasedness:** the mutation operator should be unbiased, unless knowledge about the problem has been gathered. This can be achieved by using the maximum entropy distribution that obeys knowledge about the problem as constraints.
- **Control:** the mutation operator should have parameters that affect the mutation strength (shape of the distribution), such that the extent of variation can be controlled. As known from theory, when approaching the optimal solution, the strength of mutation must be weakened steadily.

In continuous search space $\mathbb{R}^d$, these principles are fulfilled by the famous Gaussian mutation operator [3], which uses the multivariate normally distributed random vector as the variation for individual. Before presenting the commonly used Gaussian mutation in algorithms of evolution strategy, basic concepts about multivariate Gaussian distribution (also called multivariate normal distribution) are given firstly.

A *d*-dimensional random vector $\mathbf{Y} = [Y_1, \cdots, Y_d]^{\mathrm{T}}$ under multivariate normal distribution with mean vector $\mathbf{m} \in \mathbb{R}^d$ and covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$ is typically written as $\mathbf{Y} \sim \mathcal{N}_d(\mathbf{m}, \mathbf{C})$, where $\mathcal{N}_d(\mathbf{m}, \mathbf{C})$ denotes a d-dimensional multivariate normal distribution in its general form. Correspondingly, the random vector $\mathbf{Y}$ has the probability density function

$$f_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{(2\pi)^{d/2}\left[\det(\mathbf{C})\right]^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{y}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\mathbf{m})\right], \tag{2.4}$$

where $\mathbf{y} \in \mathbf{R}^d$ is the realization of the random vector $\mathbf{Y}$ and $\det(\mathbf{C})$ is the determinant of the covariance matrix $\mathbf{C}$. In mathematical equations, $\mathcal{N}_d(\mathbf{m}, \mathbf{C})$ is sometimes used like a vector which is actually sampled according the distribution given. Since the covariance matrix $\mathbf{C}$ is symmetric and positive definite, the following eigendecomposition exists:

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^{\mathrm{T}} \quad \text{and} \quad \mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^{\mathrm{T}}, \tag{2.5}$$

where $\mathbf{B}$ is an orthogonal matrix (i.e., $\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{B}\mathbf{B}^{\mathrm{T}} = \mathbf{I}$ where $\mathbf{I}$ is identity matrix), the columns of which are the eigenvectors of $\mathbf{C}$, $\mathbf{D}$ is a diagonal matrix with square roots of eigenvalues of $\mathbf{C}$ as diagonal elements, and $\mathbf{C}^{\frac{1}{2}} \in \mathbb{R}^{d \times d}$ is the symmetric square root matrix of $\mathbf{C}$ such that $\mathbf{C} = \mathbf{C}^{\frac{1}{2}}\left(\mathbf{C}^{\frac{1}{2}}\right)^{\mathrm{T}} = \mathbf{C}^{\frac{1}{2}}\mathbf{C}^{\frac{1}{2}}$. With above eigendecomposition and considering the properties of multivariate normal random vector [35], the multivariate normal distribution $\mathcal{N}_d(\mathbf{m}, \mathbf{C})$ can be written as

$$\begin{aligned} \mathcal{N}_d(\mathbf{m}, \mathbf{C}) &\sim \mathbf{m} + \mathcal{N}_d(\mathbf{0}, \mathbf{C}) \\ &\sim \mathbf{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}_d(\mathbf{0}, \mathbf{I}), \end{aligned} \tag{2.6}$$

where $\mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^{\mathrm{T}}$ is given in Equation (2.5).

In evolution strategies, the Gaussian mutation operator generates a mutated individual $\mathbf{x}'$ from an individual $\mathbf{x} \in \mathbb{R}^d$ by adding a multivariate normally distributed vector on $\mathbf{x}$. This can be expressed generally as

$$\mathbf{x}' = \mathbf{x} + \mathcal{N}_d(\mathbf{0}, \mathbf{C}) = \mathbf{x} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}_d(\mathbf{0}, \mathbf{I}). \tag{2.7}$$

Based on multivariate normal distribution, the three different mutation operators that are commonly used in evolution strategies are:

(a) **Isotropic or Spherical Gaussian mutation** generates the mutation $\mathbf{x}'$ from $\mathbf{x}$ by using $\mathbf{B} = \mathbf{I}$ and $\mathbf{D} = \sigma\mathbf{I}$ for matrices $\mathbf{B}$ and $\mathbf{D}$ in Equation (2.7), where σ is called step-size, that is,

$$\mathbf{x}' = \mathbf{x} + \sigma\mathcal{N}_d(\mathbf{0}, \mathbf{I}). \tag{2.8}$$

This corresponds with spheres with individual radii defined by σ, as indicated in Figure 2.1 (a).

(b) **Anisotropic or Axis-parallel Gaussian mutation** generates the mutation $\mathbf{x}'$ from $\mathbf{x}$ by using $\mathbf{B}=\mathbf{I}$ and $\mathbf{D}=\mathrm{diag}(\boldsymbol{\delta})=\mathrm{diag}(\delta_1,\cdots,\delta_d)$ be a diagonal matrix with different entries on the main diagonal, i.e.,

$$\mathbf{x}'=\mathbf{x}+\mathbf{I}\,\mathrm{diag}(\boldsymbol{\delta})\,\mathcal{N}_d(\mathbf{0},\mathbf{I})=\mathbf{x}+\mathcal{N}_d\left(\mathbf{0},\mathrm{diag}\left(\boldsymbol{\delta}^2\right)\right) \tag{2.9}$$

This turns the spheres into anisotropic ellipsoids with main axes parallel to the coordinate axes, as shown in Figure 2.1 (b).

(c) **Correlated Gaussian mutation** denotes the situation that matrix $\mathbf{B}$ is not just an identity matrix and $\mathbf{D}=\mathrm{diag}(\boldsymbol{\delta})=\mathrm{diag}(\delta_1,\cdots,\delta_d)$ is a diagonal matrix, i.e.,

$$\mathbf{x}'=\mathbf{x}+\mathbf{B}\,\mathrm{diag}(\boldsymbol{\delta})\,\mathcal{N}_d(\mathbf{0},\mathbf{I})=\mathbf{x}+\mathbf{B}\mathcal{N}_d\left(\mathbf{0},\mathrm{diag}\left(\boldsymbol{\delta}^2\right)\right)=\mathbf{x}+\mathcal{N}_d(\mathbf{0},\mathbf{C}) \tag{2.10}$$

This rotates the hyperellipsoids with respect to the coordinate axes, as shown in the right part of Figure 2.1 (c).

The choice of the mutation operator from above described three cases has a direct impact on the complexity of the endogenous parameters that control the multivariate normal distribution. For a problem of *d*-dimensional search space, the number of endogenous strategy parameter in case of Equation (2.8) is $O(1)$, i.e., constant. In case of Equation (2.9), a vector of endogenous parameters with size $O(d)$ is required. To adapt an arbitrary covariance matrix $\mathbf{C}$ in Equation (2.10), $O\left(d^2\right)$ endogenous parameters are required [3].

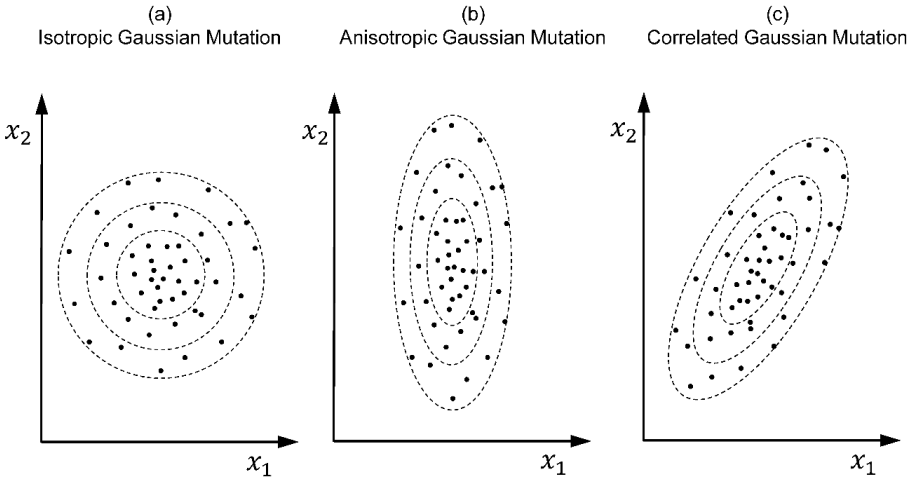

Figure 2.1 Gaussian mutation operators in evolution strategies

2.3 Parameter Control and Algorithms

Controlling the parameters of mutation operator is essential to the design of evolution strategies. Consider an evolution strategy using the isotropic mutation operator, where the step-size σ is a scaling factor for the random vector perturbation. The step-size σ controls to a large extent the convergence speed. In situations where larger step-sizes lead to larger expected improvements, a step-size control technique should aim at increasing the step-size (and decreasing it in the opposite scenario) [31]. Generally, the goal of parameter control is to drive the endogenous strategy parameters close to their optimal values. The research on ESs mainly focuses on question of adaptation of parameters of mutation operators. Different parameter control strategies bring about different algorithms for evolution strategies. In the following, three specific ESs are outlined, each of them representing an important achievement in parameter control.

2.3.1 The 1/5th Success Rule

The 1/5th success rule is a basic step-size control strategy for evolution strategies. It is based on an important discovery made by Rechenberg [36] in early research of evolution strategies. The (1+1)-ES using 1/5th success rule is the first evolution strategy and the basis of the evolutionary algorithm in the field of ES. This algorithm is implemented in Algorithm 2.1.

As the first and simplest evolution strategy, (1+1)-ES only use mutation and selection operators to guide the search. We denotes a generation (iteration) of the search by the generation counter $g \in \mathbb{N}$ ($g = 0,1,2,\cdots$). In generation g , a single offspring $\mathbf{x}_1 \in \mathbb{R}^d$ is generated from a single parent individual $\mathbf{m}^{(g)} \in \mathbb{R}^d$ by isotropic mutation i.e., $\mathbf{x}_1 = \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}_d \left(\mathbf{0}, \mathbf{I} \right)$ (Line 4 in Algorithm 2.1). If the offspring has higher fitness (lower fitness function value) than its parent, it becomes the new parent; otherwise, the parent remains (Lines 5~9 of the algorithm). The loop is terminated until termination criterion is satisfied.

Algorithm 2.1 The $(1+1)$-ES with 1/5th Success Rule

1: **given:** $d \in \mathbb{N}_+$, $d_\sigma \approx \sqrt{1+d}$
2: **initialize** $\mathbf{m}^{(0)} \in \mathbb{R}^d$, $\sigma^{(0)} > 0$, $g \leftarrow 0$
3: **repeat**
4: $\quad \mathbf{x}_1 = \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}_d(\mathbf{0}, \mathbf{I})$ //mutation
5: $\quad$ **if** $f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})$ **then**
6: $\quad\quad \mathbf{m}^{(g+1)} = \mathbf{x}_1$ //selection of $\mathbf{x}_1$ as the new parent if it is better than $\mathbf{m}^{(t)}$
7: $\quad$ **else**
8: $\quad\quad \mathbf{m}^{(g+1)} = \mathbf{m}^{(g)}$
9: $\quad$ **end if**
10: $\quad \sigma^{(g+1)} \leftarrow \sigma^{(g)} \exp\left[\frac{1}{d_\sigma}\left(\mathbb{E}\left[\mathbb{I}\left(f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})\right)\right] - \frac{1}{5}\right)\right]$ //step-size update
11: $\quad g \leftarrow g+1$
12: **until** termination criterion is fulfilled

In (1+1)-ES, the adaptation of the step-size, which is the only one strategy parameter of (1+1)-ES, is of crucial importance for reliable results and efficiency of the algorithm. The step-size significantly affects the convergence speed. The step-size has to be adapted in order to speed up the optimization process. Usually, in situations where larger step-size lead to larger expected improvements, a step-size control technique should aim at increasing the step-size, and decrease it in the opposite situations. The step-size σ in (1+1)-ES is adapted according to the 1/5th success rule.

The idea of the 1/5th success rule is to increase the mutation rate (step-size in isotropic mutation), if the success probability (i.e., the empirical probability that $f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})$), which refers to the ratio between successful mutations and all mutations, is larger, and to decrease it, if the success probability is smaller. Specifically, if about 1/5 of all mutations are successful, the step-size is optimal and no adaptation is required. The step-size needs to be reduced, if the success rate is smaller than 1/5, and needs to be increased, if the success rate is larger than 1/5. The new step-size $\sigma^{(g+1)}$ is updated according to Line 10 in Algorithm 2.1. In the algorithm, the empirical probability that $f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})$ is expressed by $\mathbb{E}\left[\mathbb{I}\left(f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})\right)\right]$, where $\mathbb{E}[\cdot]$ denotes the expected value function, and $\mathbb{I}(\cdot)$ is the indicator function such that $\mathbb{I}\left(f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})\right) = 1$ only if $f(\mathbf{x}_1) \le f(\mathbf{m}^{(g)})$ and

$\mathbb{I}\left(f\left(\mathbf{x}_1\right) \leq f\left(\mathbf{m}^{(g)}\right)\right) = 0$ otherwise. The success probability information is obtained from previous iterations.

2.3.2 Cumulative Step-Size Adaptation (CSA)

The Cumulative Step-Size Adaptation Evolution Strategy (CSA-ES) developed by Ostermeier et al. [37] adapts the step-size σ by using a so-called evolution path that records information accumulated from the preceding generations instead of using information from the current generation only. The evolution path records the sum of consecutive successful mutation steps to make a decision about possible corrections of the step-size. This information can improve the adaptation and search procedure remarkably [31]. When successful steps are towards the same direction, the evolution path which records the sum of mutation steps will be relatively long in this direction. Because the same distance in this direction can be covered with larger steps, the step-size should be increased [38]. On the contrary, if the orientations of successful steps are opposite to each other, they sum up making the evolution path relatively short. In this case, the used step-size is too large and should be decreased.

Algorithm 2.2 outlines the $(\mu/\mu_I, \lambda)$-ES with Cumulative Step-Size Adaptation (CSA). In this algorithm, comma selection, isotropic mutation and intermediate recombination are used. The step-size σ is adapted by using the evolution path $\mathbf{p}_\sigma$. In the evolution loop, at generation g, λ offspring $(\mathbf{x}_1, \cdots, \mathbf{x}_\lambda)$ are generated by adding a isotropic Gaussian mutation on the solution vector $\mathbf{m}^{(g)}$ (also known as the mean of mutation distribution), which is described in Lines 4~6. The fitness of these λ individuals then are evaluated (Line 7). The selection operator chooses μ best individuals to form the new parent population according to their fitness values. Then, the intermediate recombination of the μ parents generates a single new solution vector $\mathbf{m}^{(g+1)}$, which is the new mean vector of the mutation distribution (Line 9).

The evolution path $\mathbf{p}_\sigma$ is updated (Line 10) using local information about a successful mutation step $\frac{1}{\mu}\sum_{i=1}^{\mu}\mathbf{z}_{i:\lambda}$, a decay factor c_σ is used to decrease the importance of previously

performed steps with time, and the factor $\sqrt{c_\sigma\left(2-c_\sigma\right)}\sqrt{\mu}$ is set to normalize the variance of $\mathbf{p}_\sigma$ such that $\mathbf{p}_\sigma^{(g+1)} \sim \mathcal{N}_d\left(\mathbf{0},\mathbf{I}\right)$. Finally, the step-size σ is updated by using the evolution path $\mathbf{p}_\sigma$ as expressed in Line 11. The step-size increases if the length of the evolution path $\left\|\mathbf{p}_\sigma^{(g+1)}\right\|$ is longer than the expected length of the evolution path under random selection $\mathbb{E}\left(\left\|\mathcal{N}_d\left(\mathbf{0},\mathbf{I}\right)\right\|\right)$, and decreases otherwise. The expectation of $\left\|\mathcal{N}_d\left(\mathbf{0},\mathbf{I}\right)\right\|$, i.e. $\mathbb{E}\left(\left\|\mathcal{N}_d\left(\mathbf{0},\mathbf{I}\right)\right\|\right)$, is approximated by $\sqrt{d}\left(1-\frac{1}{4d}+\frac{1}{21d^2}\right)$. The damping factor d_σ is used to control the change of the step-size.

Algorithm 2.2 The $\left(\mu/\mu_I,\lambda\right)$-ES with Cumulative Step-Size Adaptation

1: **given:** $d \in \mathbb{N}_+$, $\lambda = 4+\left\lfloor 3\ln\left(d\right)\right\rfloor$, $\mu = \left\lfloor \lambda/2 \right\rfloor$,
$c_\sigma = \left(\mu+2\right)/\left(d+\mu+5\right)$, $d_\sigma = 1+2\max\left(0,\sqrt{\left(\mu-1\right)/\left(d+1\right)}-1\right)+c_\sigma$.
2: **initialize** $\mathbf{m}^{(0)} \in \mathbb{R}^d$, $\sigma^{(0)} > 0$, $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$, $g \leftarrow 0$
3: **repeat**
4: **for** $k = 1,\cdots,\lambda$ **do**
5: $\mathbf{z}_k = \mathcal{N}_d\left(\mathbf{0},\mathbf{I}\right)$ //i.i.d. for each $\mathbf{z}_k$
6: $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\mathbf{z}_k$ //mutation
7: $f_k = f\left(\mathbf{x}_k\right)$
8: **end for**
9: $\mathbf{m}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\mathbf{x}_{i:\lambda}$ //intermediate recombination
10: $\mathbf{p}_\sigma^{(g+1)} \leftarrow \left(1-c_\sigma\right)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma\left(2-c_\sigma\right)}\frac{\sqrt{\mu}}{\mu}\sum_{i=1}^{\mu}\mathbf{z}_{i:\lambda}\mathbf{m}^{(g+1)}$ //update evolution path
11: $\sigma^{(g+1)} \leftarrow \sigma^{(g)}\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\mathbb{E}\left(\left\|\mathcal{N}\left(\mathbf{0},\mathbf{I}\right)\right\|\right)}-1\right)\right)$ //step-size update
12: $g \leftarrow g+1$
13: **until** termination criterion is fulfilled

2.3.3 Covariance Matrix Adaptation (CMA)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [39, 40], is a highly developed evolution strategy and has become a standard for continuous black-box evolutionary optimization. It is a powerful optimization algorithm and performs especially well in non-smooth, multimodal back-box problems. The CMA-ES adopts the correlated mutation operator, which makes it a high-level algorithm compared with previous described

algorithms that use isotropic mutation. In CMA-ES, two techniques, namely the covariance matrix adaptation (CMA) and the cumulative step-size adaptation (CSA), are used for adapting the covariance matrix of mutation and the step-size, respectively.

The $(\mu/\mu_w,\lambda)$-CMA-ES is detailed in the remaining part of this subsection. Briefly speaking, in generation loop of the $(\mu/\mu_w,\lambda)$-CMA-ES (simply denoted as CMA-ES hereafter), λ offspring are generated by mutation based on a single solution vector (the mean vector of mutation distribution), comma selection is then performed to select μ ($\mu \le \lambda$) best individuals according to their fitness, the weighted recombination of selected individuals creates the single new solution. The generation loop stops until the termination criterion is fulfilled. This generation loop of CMA-ES is presented in details in the following.

Mutation or Sampling

In $(\mu/\mu_w,\lambda)$-CMA-ES, the λ offspring are generated by correlated mutation from the same single solution vector, i.e.,

$$\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}_d\left(\mathbf{0},\mathbf{C}^{(g)}\right) = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_k, \quad \text{for } k=1,\cdots,\lambda, \tag{2.11}$$

where: $g \in \mathbb{N}^0$ is the counter of generation.

λ is the number of offspring. Its default value is $\lambda = 4+\lfloor 3\ln(d)\rfloor$ (d is the dimension of the problem).

$\mathbf{x}_k \in \mathbb{R}^d$ is the k-th offspring (individual, search point, solution).

$\mathbf{m}^{(g)} \in \mathbb{R}^d$ is the mean of mutation distribution (a solution vector), which is formed by weighted recombination of selected μ best individuals after initialization. Its initial value $\mathbf{m}^{(0)}$ is given by the user.

$\sigma^{(g)} \in \mathbb{R}^+$ denotes the step-size (overall standard deviation) at generation g. Its initial values $\sigma^{(0)}$ is given according to the bounds of search space.

$\mathbf{C}^{(g)} \in \mathbb{R}^{d\times d}$ is a symmetric positive definite covariance matrix at generation g. Its default initial value is $\mathbf{C}^{(0)} = \mathbf{I}$. $\left(\mathbf{C}^{(t)}\right)^{\frac{1}{2}} \in \mathbb{R}^{d\times d}$ is the symmetric square root matrix

of $\mathbf{C}^{(g)}$ such that $\mathbf{C}^{(g)} = \left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\left[\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\right]^{\mathrm{T}} = \left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}$, which can be obtained according to Equation (2.5)

$\mathbf{z}_k \in \mathbb{R}^d$ are random vectors (mutation vector) sampled from $\mathcal{N}_d(\mathbf{0},\mathbf{I})$, i.e. $\mathbf{z}_k = \mathcal{N}_d(\mathbf{0},\mathbf{I})$.

From Equation (2.11), the λ offspring also can be viewed as sampling from a multivariate normal distribution with mean vector $\mathbf{m}^{(g)}$ and covariance matrix $\left(\sigma^{(g)}\right)^2\mathbf{C}^{(g)}$, i.e., $\mathbf{x}_k \sim \mathcal{N}_d\left(\mathbf{m}^{(g)}, \left(\sigma^{(g)}\right)^2\mathbf{C}^{(g)}\right)$. To define the complete iteration procedure, the remaining question is how to calculate $\mathbf{m}^{(g+1)}$, $\mathbf{C}^{(g+1)}$ and $\sigma^{(g+1)}$ for the next generation $g+1$.

Selection and Recombination

After generating λ offspring, their fitness function values are evaluated. According to their fitness, μ ($\mu \le \lambda$) best individuals from λ offspring are selected to become the parent population and undergo weighted recombination to generate a single new solution vector $\mathbf{m}^{(g+1)}$. Thus, the new solution $\mathbf{m}^{(g+1)}$ is computed as a weighted average of the selected points:

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{u} w_i \mathbf{x}_{i:\lambda} = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}} \sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda} \tag{2.12}$$

$$\sum\nolimits_{i=1}^{\mu} w_i = 1, \qquad w_1 \ge w_2 \ge \cdots \ge w_\mu > 0 \tag{2.13}$$

where: μ is the number of selected individuals. Its default value is $\mu = \lfloor \lambda/2 \rfloor$.

$w_i, i = 1,\cdots,\mu$ are positive weights for recombination. When $w_i = 1/\mu, i = 1,\cdots,\mu$, it corresponds to the intermediate recombination. In the standard CMA-ES, the super-linear decrease weights are used, that is
$w_i = \left[\ln(\lambda/2+1/2) - \ln(i)\right] \Big/ \sum_{j=1}^{\mu}\left[\ln(\lambda/2+1/2) - \ln(j)\right],\ i = 1,\cdots,\mu$.

$\mathbf{x}_{i:\lambda}$ signifies the *i*-th best individual out of $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\lambda$ from Equation (2.11). The index $i:\lambda$ denotes the index of the *i*-th ranked individual and $f(\mathbf{x}_{1:\lambda}) \le f(\mathbf{x}_{2:\lambda}) \le \cdots \le f(\mathbf{x}_{\lambda:\lambda})$ where f is the objective function.

$\mathbf{z}_{i:\lambda}$ is the mutation vector associated with $\mathbf{x}_{i:\lambda}$.

Adaptation of Step-size and Covariance Matrix

The CMA-ES uses accumulative step-size adaptation (CSA) to adapt the step-size σ and the covariance matrix adaptation (CMA) to update the covariance matrix $\mathbf{C}$. These two techniques (CSA and CMA) use so-called evolution paths for accumulating strategy parameter information across several generations [3]. Two evolution paths, $\mathbf{p}_\sigma \in \mathbb{R}^d$ for adaptation of step-size σ and $\mathbf{p}_c \in \mathbb{R}^d$ for covariance matrix $\mathbf{C}$ adaptation, are used in CMA-ES.

The evolution path $\mathbf{p}_\sigma$ and the step-size σ are recursively computed [3, 41] according to the equations below:

$$\begin{aligned}\mathbf{p}_\sigma^{(g+1)} &= (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_w}\left(\mathbf{C}^{(t)}\right)^{-\frac{1}{2}}\frac{\mathbf{m}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(t)}} \\ &= (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda}\end{aligned} \tag{2.14}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right)}-1\right)\right) \tag{2.15}$$

where: $\mathbf{p}_\sigma^{(g)} \in \mathbb{R}^d$ is the evolution path for global step-size at generation g, its default initial value is $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$.

$c_\sigma \in [0,1]$ is the time constant for the adaptation of the step-size, its default value [41] is $c_\sigma = (\mu_w+2)/(d+\mu_w+5)$.

μ_w is the so-called variance effective selection mass, which is defined by: $\mu_w = \left(\sum_{i=1}^{\mu} w_i\right)^2 \Big/ \sum_{i=1}^{\mu} w_i^2$. If equal weights $w_i = 1/\mu$ are used, μ_w is equal to μ.

$\left(\mathbf{C}^{(g)}\right)^{-\frac{1}{2}}$ is the inverse of $\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}$ which is the symmetric square root matrix of $\mathbf{C}^{(g)}$.

d_σ is the damping factor, its default value is $d_\sigma = 1+2\max\left(0,\sqrt{\frac{\mu_w-1}{d+1}}-1\right)+c_\sigma$.

$\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right)$ is the expectation of the Euclidean norm of a $\mathcal{N}_d(\mathbf{0},\mathbf{I})$ distributed random vector, which can be approximated by

$$\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right) = \sqrt{2}\Gamma\left(\frac{d+1}{2}\right)\Big/\Gamma\left(\frac{d}{2}\right) \approx \sqrt{d}\left(1-\frac{1}{4d}+\frac{1}{21d^2}\right)$$

The evolution path $\mathbf{p}_\mathrm{c}$ and covariance matrix $\mathbf{C}$ are updated [3, 41] according to the equations below:

$$\begin{aligned}\mathbf{p}_\mathrm{c}^{(g+1)} &= (1-c_\mathrm{c})\mathbf{p}_\mathrm{c}^{(g)} + h_\sigma\sqrt{c_\mathrm{c}(2-c_\mathrm{c})}\sqrt{\mu_w}\,\frac{\mathbf{m}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(t)}} \\ &= (1-c_\mathrm{c})\mathbf{p}_\mathrm{c}^{(g)} + h_\sigma\sqrt{c_\mathrm{c}(2-c_\mathrm{c})}\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\end{aligned} \tag{2.16}$$

$$\begin{aligned}\mathbf{C}^{(g+1)} &= (1-c_1-c_u)\mathbf{C}^{(g)} + c_1\left(\mathbf{p}_\mathrm{c}^{(g+1)}\left(\mathbf{p}_\mathrm{c}^{(g+1)}\right)^{\mathrm{T}} + \delta(h_\sigma)\mathbf{C}^{(g)}\right) \\ &\quad + c_\mu\sum_{i=1}^{u} w_i\left(\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\right)\left(\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\right)^{\mathrm{T}}\end{aligned} \tag{2.17}$$

where: $\mathbf{p}_\mathrm{c}^{(g)} \in \mathbb{R}^d$ is the evolution path for covariance matrix at generation g, its default initial value is $\mathbf{p}_\mathrm{c}^{(0)} = \mathbf{0}$.

$c_\mathrm{c} \in [0,1]$ is the constant for covariance matrix adaptation. $1/c_\mathrm{c}$ is the backward time horizon of the evolution path $\mathbf{p}_\mathrm{c}$. The default value of c_c is:

$c_c = (4+\mu_w/d)/(d+4+2\mu_w/d)$

h_σ is a Heaviside function defined by:

$$h_\sigma = \begin{cases} 1 & \text{if } \dfrac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\sqrt{1-(1-c_\sigma)^{2(g+1)}}} < \left(1.4+\dfrac{2}{d+1}\right)\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right) \\ 0 & \text{otherwise}\end{cases}$$

The purpose of function h_σ is to avoid an update of $\mathbf{p}_\mathrm{c}$ to take information of the current generation into account, when $\left\|\mathbf{p}_\sigma^{(g+1)}\right\|$ is too large [42]. This prevents a too fast increase of axes of $\mathbf{C}$ when the step-size is far too small.

c_1 and c_μ are the learning rate for rank-one-update and rank-μ-update, respectively. Their default values are:

$c_1 = \dfrac{2}{(d+1.3)^2+\mu_w}$, and $c_u = \min\left(1-c_1, 2\dfrac{\mu_w-2+1/\mu_w}{(d+2)^2+\alpha_u\,\mu_w/2}\right)$ with $\alpha_\mu = 2$.

$\delta(h_\sigma) = (1-h_\sigma)c_\mathrm{c}(2-c_\mathrm{c})$ is of minor relevance. In the unusual case of $h_\sigma = 0$, it substitutes for the second summand from Equation (2.16) in (2.17) [41].

In summary, for CMA-ES iteration, the new offspring population of search points is generated by Equation (2.11), the selection and recombination is defined by Equation (2.12), the adaptation of step-size is given by Equations (2.14) and (2.15), and the covariance matrix is updated with Equations (2.16) and (2.17). The iteration defined above is repeatedly executed until the termination criterion fulfilled. Putting it all together, the pseudocode of $(\mu/\mu_w,\lambda)$-CMA-ES is given in Algorithm 2.3.

Algorithm 2.3 The $(\mu/\mu_w,\lambda)$-CMA-ES

given: $d \in \mathbb{N}_+$, $\lambda = 4+\lfloor 3\ln(d)\rfloor$, $\mu = \lfloor \lambda/2 \rfloor$,

$$w_i = \frac{\ln(\lambda/2+1/2)-\ln(i)}{\sum_{j=1}^{\mu}\left[\ln(\lambda/2+1/2)-\ln(j)\right]} \text{ for } i=1,\cdots,\mu,\ \ \mu_w = \left(\sum_{i=1}^{\mu} w_i\right)^2 \Big/ \sum_{i=1}^{\mu} w_i^2,$$

$$c_\sigma = \frac{\mu_w+2}{d+\mu_w+5},\ d_\sigma = 1+2\max\left(0,\sqrt{\frac{\mu_w-1}{d+1}}-1\right)+c_\sigma$$

$$c_c = \frac{4+\mu_w/d}{d+4+2\mu_w/d},\ c_1 = \frac{2}{(d+1.3)^2+\mu_w},\ c_u = \min\left(1-c_1, 2\frac{\mu_w-2+1/\mu_w}{(d+2)^2+\mu_w}\right).$$

initialize $\mathbf{m}^{(0)} \in \mathbb{R}^d$, $\sigma^{(0)} > 0$, $\mathbf{p}_c^{(0)} = \mathbf{0}$, $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$, $\mathbf{C}^{(0)} = \mathbf{I}$, $g \leftarrow 0$

repeat

  for $k = 1,\cdots,\lambda$ **do**

    $\mathbf{z}_k = \mathcal{N}_d(\mathbf{0},\mathbf{I})$ //i.i.d. for each $\mathbf{z}_k$

    $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{1/2}\mathbf{z}_k$ //mutation

    $f_k = f(\mathbf{x}_k)$

  end for

  $\mathbf{m}^{(g+1)} \leftarrow \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda}$

  $\mathbf{p}_\sigma^{(g+1)} \leftarrow (1-c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda}$

  $\sigma^{(g+1)} \leftarrow \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right)}-1\right)\right)$

  $\mathbf{p}_c^{(g+1)} \leftarrow (1-c_c)\mathbf{p}_c^{(g)} + h_\sigma\sqrt{c_c(2-c_c)}\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i \left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}$

  $\mathbf{C}^{(g+1)} \leftarrow (1-c_1-c_u)\mathbf{C}^{(g)} + c_1\left(\mathbf{p}_c^{(g+1)}\left(\mathbf{p}_c^{(g+1)}\right)^{\mathrm{T}} + \delta(h_\sigma)\mathbf{C}^{(g)}\right) + c_\mu\sum_{i=1}^{u} w_i \left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\left(\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\right)^{\mathrm{T}}$

  $g \leftarrow g+1$

until termination criterion is fulfilled

3. Surrogate Modeling of Computer Experiments

This chapter reviews the surrogate modeling method, which is an approach to model the (input-output) behavior of the simulation model. Computer simulation tools, such as finite element analysis (FEA) and computational fluid dynamics (CFD), are more and more widely applied in engineering problems. Many engineering design problems require a number of simulations to evaluate design objective. However, for many read world problems, a single simulation can take many minutes, hours, or even days to complete. Consequently, computation-intensive tasks, such as design optimization, sensitivity analysis and reliability analysis, become impractical or impossible because they require doing hundreds, thousands or even millions of simulations. One way of reducing this computational burden is by constructing approximation models, known as surrogate models, or metamodels, that approximate the input-output relationship of the simulation model as closely as possible while being computationally cheap to evaluate [43]. Then, many tasks can be implemented by using the built surrogate model. Computational cost is thus apparently reduced owing to the aid of cheap-evaluated surrogate model.

Main issues about surrogate modeling are presented in the following of this chapter. In Section 3.1, the general process of surrogate modeling is introduced. The three steps of surrogate modeling, (i) design of experiments; (ii) training of the surrogate model; and (iii) model validation, are respectively addressed in Section 3.2, 3.3 and 3.4.

3.1 Surrogate Modeling Process

Surrogate modeling is concerned with the construction of mathematical models to describe the relationships between specific inputs and outputs exhibited by the simulation model, based on a set of limited data acquired by running the simulation model with

intelligently chosen input variables. Since simulation models usually generate multiple (a set of) responses (outputs), a surrogate model can be constructed for each of the outputs of interest. Without loss of generality, only one response of interest is considered in this chapter. Let

$$y = f(\mathbf{x}) = f(x_1, \cdots, x_d), \quad \mathbf{x} = [x_1, \cdots, x_d]^{\mathrm{T}} \in S \subset \mathbb{R}^d, \tag{3.1}$$

denotes a simulation model which models a physical system, where $\mathbf{x} = [x_1, \cdots, x_d]^{\mathrm{T}}$ is a vector of d design variables or input variables (input), y is a response (output), S is referred to as the design space or input variable space, and the function f signifies the computer simulation model, such as FEA and CFD model of a problem, that maps the d-dimensional inputs into a scalar output. Model (3.1) generally can be regarded as a solution of a set of equations, including linear, nonlinear, ordinary and/or partial differential equations, and it is often impossible to obtain an analytic solution for the equations.

Then, the general surrogate modeling problem can be stated as follows: "Given a set of samples (observations) $\mathbf{X} = \left[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\right]^{\mathrm{T}}$ in the design space and the corresponding response values $\mathbf{y} = \left[y^{(1)}, \cdots, y^{(n)}\right]^{\mathrm{T}} = \left[f\left(\mathbf{x}^{(1)}\right), \cdots, f\left(\mathbf{x}^{(n)}\right)\right]^{\mathrm{T}}$, the goal is to obtain an approximation model $y = \hat{f}(\mathbf{x})$ that adequately represents the input-output relationship exhibited by simulation model over a given design domain." The set of sampling points $\mathbf{X}$ is determined by design of experiments (DOE), and the corresponding responses $\mathbf{y}$ are generated by the experiments conducted under that DOE. Thus, design of experiments is the first step in the construction of surrogate models. The samples (sample points) used to construct the surrogate models are generally called training points. The set of data that includes input data and the corresponding output data is called training data set, while the construction of the surrogate models based on training data set is often called model training. With the training data set, a variety of surrogate models can be trained. Before the trained surrogate model is used in lieu of the original simulation model, it is necessary to evaluate the performance or expected accuracy of the surrogate model.

Therefore, the typical process of surrogate modeling mainly involves three steps (i) design of experiments; (ii) construction or training of the surrogate model; and (iii) model

validation. This process is illustrated in Figure 3.1. These three issues for surrogate modeling are discussed separately in following sections.

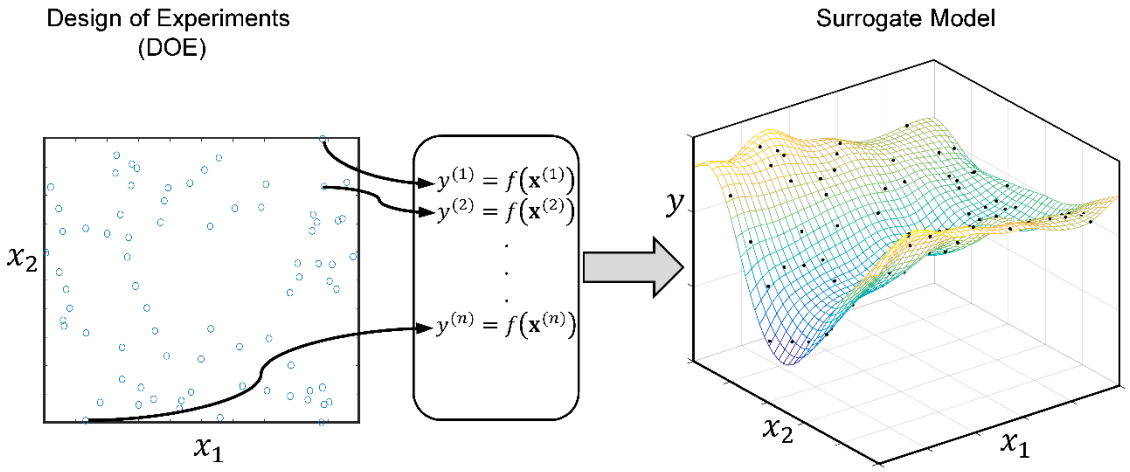

Figure 3.1 Process of surrogate modeling

3.2 Design of Experiments

Design of experiments (DOE, or experimental design) refers to the techniques that are used for guiding the choice of experiments to be performed in an efficient way. DOE techniques were originally developed to study the behavior of systems through physical experiments [44]. The basic objective of DOE is to determine multiple combinations of the controlled parameters (or conditions) at which the experiments will be conducted. Each combination of controlled parameters, in mathematical terms, can be considered a sample. Although DOE originally referred to physical experiments, in modern times, it would includes both physical and computer experiments. This section only focus on the design of computer experiments.

In computer experiments, DOE is a strategy for allocating samples (points) in the design space that aims at maximizing the amount of information acquired [4]. Computer simulations (experiments) are performed at these points to create the training data set that is subsequently used to construct the surrogate model. Obviously, there is a trade-off between the number of samples and the amount of information that can be extracted from these samples. Thus, a

good experimental design should minimize the number of samples needed to acquire enough information.

In following part of this section, the factorial designs, which are usually referred to as classical DOE techniques, and several space-filling or modern DOE techniques are presented. Before giving descriptions of these DOE techniques, some terminology in design of computer experiments are introduced. The controllable variables that are of interest in computer experiments are usually called design variables or input variables or simply called inputs. Design space or input variable space is the space where the input variables take values. A point in design space or an input setting, which is a combination of input variables, is called a sample (sample point). A run is the implementation of an experiment (computer simulation) with the input variables given in a sample. Output(s) or response(s) is the result of a run based on the purpose of the experiment.

3.2.1 Factorial Designs

Factorial designs, which are classical DOE techniques [45], are straightforward techniques for experimental design. In factorial designs, the range of each design variable is divided into different levels between the upper and lower limits of the design space. Factorial designs allocate samples at combinations of different levels of all the design variables. In other words, a factorial design is a set of level-combinations of the factors. A factorial design is called symmetric if all factors have the same number of levels ; otherwise, it is called asymmetric.

When samples are located at all the combinations of the different levels of all the design variables, the design is called full factorial design. Obviously, the number of sample points in a full factorial design should be $n = \prod_{j=1}^{d} q_j$, where q_j is the number of levels of the design variable j . When all the design variables have the same number of levels q , the number of samples are $n = q^d$. Figure 3.2 illustrates a full factorial design of three-dimensional design space with three level for all variables. The number of sample points of full factorial design increases exponentially with the number of design variables. Thus, in high-dimensional

problems, the full factorial design approach may be cost/time prohibitive. Therefore, we consider implementing a subset of all the level-combinations that have a good representation of the complete combinations.

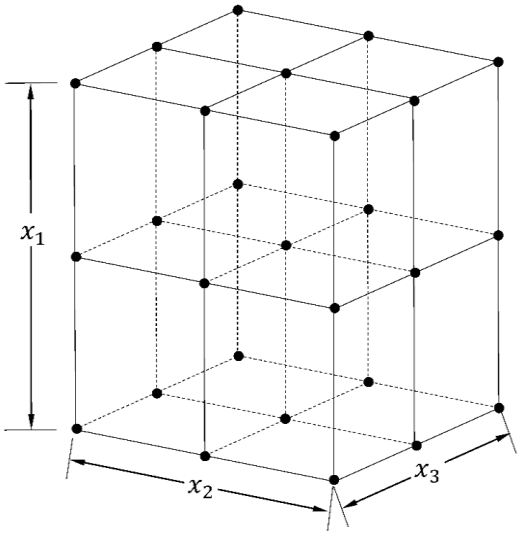

Figure 3.2 A 3-level and 3-dimensional full factorial design (27 points)

In situations that the running of an experiment is expensive and the number of design variables is large, fractional factorial design (FFD) can be used. Fractional factorial design (FFD) uses a fraction of the full factorial design sample points, i.e., a FFD is a subset of all level-combinations of the design variables. Consequently, how to choose a good subset is the most important issue in FFD. A carefully selected combination known as the orthogonal array is recommended in the literature and has been widely used in practice. In practice, other alternative factorial designs, such as central composite design, star design, and Box-Behnken design, can be used.

3.2.2 Latin Hypercube Designs

This subsection discusses some modern DOE techniques [45] for planning computer experiments, which tend to allocate samples throughout the design space as uniformly as possible. Such designs are broadly referred to as space-filling designs.

Because of the deterministic feature of computer experiments, i.e., samples with the same input setting will yield identical outputs, the basic principles of experimental design for controlling noise and bias, replication, blocking, and randomization, are irrelevant to computer experiments [46]. The exact functional form of the relationship between inputs and the response is unknown and often very complicated, although the response can be computed at any given inputs. Various surrogate models can be built using different techniques. However, before data are collected, quite often little priori or background knowledge is available about which model would be appropriate, and designs for computer experiments should facilitate diverse modeling methods [47]. For this purpose, a space-filling design is the best choice. The design region in which to make prediction may be unspecified at the data collection stage. Therefore, it is appropriate to use designs that represent all portions of the design space. When the primary goal of experiments is to make prediction at unsampled points, space-filling designs allow us to build a predictor with better average accuracy.

A variety of space-filling designs are available. One of the most popular category of space-filling designs techniques is the Latin hypercube design, which is based on Latin hypercube sampling. Basic Latin hypercube design and its variants are presented in the following.

Basic Latin Hypercube Designs

Designs generated by Latin hypercube sampling are called Latin hypercube designs (LHD) in computer experiments design. Latin hypercube sampling (LHS) was proposed by McKay et al. (1979) specifically for computer experiments in which inputs are chosen randomly from some specified distribution. Latin hypercube designs have one-dimensional uniformity in that, when projected on each dimension, each interval of the dimension has exactly one observation.

In order to allocate n sample points in d-dimensional design space with LHS, the range of each variable is firstly divided into n on-overlapping intervals with equal probability, which yields a total number of n^d bins in the design space. The samples are randomly

selected in the design space so that (i) each sample is randomly placed inside a bin, and (ii) for all one-dimensional projections of the n samples and bins, there is exactly one sample in each bin [4]. Figure 3.3 shows a LHS realization of 10 samples for two design variables.

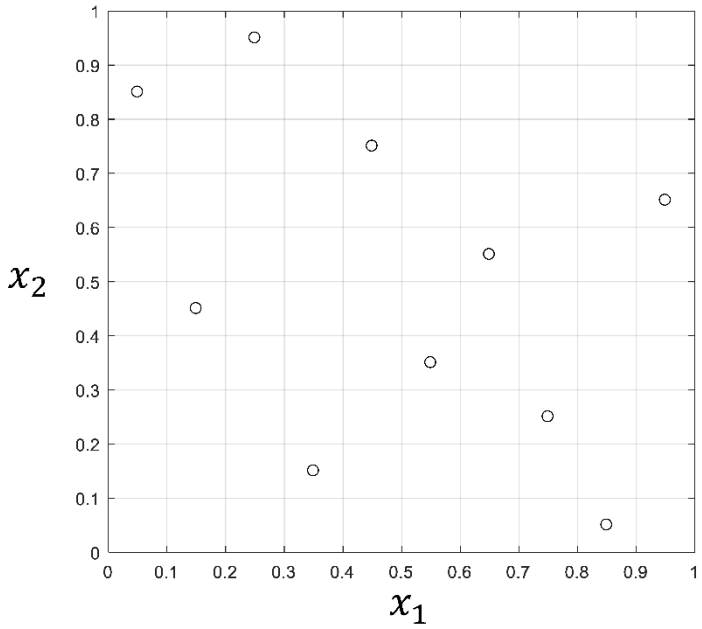

Figure 3.3 LHS realization of 10 samples in a two-dimensional design space

Mathematically, the algorithm for generating LHS samples in design space $[0,1]^d$ is

$$x_j^{(i)} = \frac{\pi_j^{(i)} - U_j^{(i)}}{n}, \quad 1 \le i \le n, 1 \le j \le d \tag{3.2}$$

where d is the number of design variables, n is the number of samples, $\pi_j^{(1)}, \cdots, \pi_j^{(n)}$ are independent uniform random permutations of the integers 1 to n, and $U_j^{(i)}$ are independent random numbers from $[0,1]$. The subscript j denotes the dimension index and the superscript (i) denotes sample number. Then the sampling plan $\mathbf{X} = \left[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\right]^{\mathrm{T}}$, where $\mathbf{x}^{(i)} = \left[x_1^{(i)}, \cdots, x_d^{(i)}\right]^{\mathrm{T}}$, is a LHD denoted by $LHD(n,d)$. If each U_{ij} in Equation (3.2) is taken to be 0.5, the corresponding LHS is called midpoint Latin hypercube sampling or centered Latin hypercube sampling. For each variable, Latin hypercube designs have exactly one point in each of the n intervals. This property is referred to as one-dimensional uniformity.

One drawback of Latin hypercube sampling is that there is more than one possible arrangement of bins and samples that meets the LHS criteria. So, a randomly generated Latin

hypercube design does not necessarily perform well with respect to criteria of uniformity. For example, samples allocated along the design space diagonal satisfy conditions of LHS, but they are non-uniformly distributed. Fortunately, there are extensions to the basic LHS technique that can improve the uniformity of Latin hypercube designs. In extensions of basic LHS, the Latin hypercube designs based on distance are of desirable properties.

Latin Hypercube Designs based on Measures of Distance

To construct space-filling Latin hypercube designs, one natural approach is to make use of distance criteria. One of the most widely-used measures to evaluate the uniformity ('space-fillingness') of a sampling plan is the maximin metric introduced by Johnson et al. (1990). The criterion based on this may be defined as follows.

Let $\mathbf{x}^{(i_1)}$ and $\mathbf{x}^{(i_2)}$ be two design points in the design space $D = [0,1]^d$. For $p > 0$, the distance between $\mathbf{x}^{(i_1)}$ and $\mathbf{x}^{(i_2)}$ is defined as

$$d\left(\mathbf{x}^{(i_1)}, \mathbf{x}^{(i_2)}\right) = \left(\sum_{j=1}^{d} \left| x_j^{(i_1)} - x_j^{(i_2)} \right|^p\right)^{1/p}. \tag{3.3}$$

When $p = 1$ and $p = 2$, the measure in Equation (3.3) becomes the rectangular and Euclidean distance, respectively. Let $d_1, \cdots, d_m$ be the list of unique values of distances between all possible pairs of points in a design $\mathbf{X}$, sorted in ascending order. Further, let $J_1, \cdots, J_m$ be defined such that J_j is the number of pairs of points in $\mathbf{X}$ separated by the distance d_j.

The simple maximin distance criterion seeks a design $\mathbf{X}$ in the design space S that maximizes the smallest inter-point distance d_1 among all available designs. This criterion attempts to place the design points such that no two points are too close to each other.

Clearly, this definition could be applied to any set of sampling plans, but, since we would like to keep the appealing stratification properties of Latin hypercube, we restrict our scope to that class of designs. Nonetheless, even across this narrower domain, Definition 1.1 might still yield several maximin designs. Therefore, we shall use the more complete 'tie-breaker' definition of Morris and Mitchell (1995). Thus, we call design $\mathbf{X}$ a maximin design among all available designs if it maximizes d_1, among designs for which this is true,

minimizes J_1, among designs for which this is true, maximizes d_2, among designs for which this is true, minimizes J_2,…, minimizes J_m.

An extended definition of a maximin design was given by Morris and Mitchell (1995). In order to rank competing designs, based on above definition, they introduced a computationally efficient scalar-value criterion

$$\Phi_q = \left(\sum_{i=1}^{m} \frac{J_i}{d_i^q} \right)^{1/q} \tag{3.4}$$

where q is a positive integer. The smaller the value of Φ_q, the better the space-filling properties of $\mathbf{X}$ will be. This scalar value distills the cumbersome definition of the maximin criterion into a rather neat and compact form, but it raises the question of how to choose the value of q in Equation (3.4). Values of q are chosen depending on the size of the design searched for, ranging from 5 for small designs to 20 for moderate-sized designs to 50 for large designs [47]. The idea of above described maximin design sounds simple and desirable. However, generating maximum Latin hypercube designs is a challenging task particularly for large designs.

3.2.3 Other Space-filling Designs

Besides Latin hypercube designs, other space-filling design techniques including orthogonal array, quasi-Monte Carlo sampling, and uniform design can be found in literature. A good summary of space-filling designs for computer experiments can be found in [45] and Chapter 5 of [48]. Sampling techniques can be improved by minimizing a specific non-uniformity measure with optimization techniques. The above described maximin Latin hypercube design is a typical instance for this.

3.3 Surrogate Models

Once the training data set has been collected from an experiment, the next step is to construct a surrogate model which describes the relationship between the inputs and output(s). Generally, the n samples in d-dimensional design space from DOE can be

represented by a $n \times d$ matrix $\mathbf{X}$ and their corresponding responses are expressed by a $n \times 1$ vector $\mathbf{y}$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(n)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} = \begin{bmatrix} f\left(\mathbf{x}^{(1)}\right) \\ \vdots \\ f\left(\mathbf{x}^{(n)}\right) \end{bmatrix} \tag{3.5}$$

where $\mathbf{x}^{(i)}, i = 1, \cdots, n$ denotes the *i*-th sample, which is a combination of d design variables, and $x_j^{(i)}, i = 1, \cdots, n,\ j = 1, \cdots, d$ denotes the *j*-th component of the *i*-th sample, and function f signifies the computer simulation model that can calculate the response with any given inputs, $y^{(i)}, i = 1, \cdots, n$ signifies the corresponding response with inputs $\mathbf{x}^{(i)}$. With the training data set $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \cdots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\}$, our goal is to construct a cheap-to-evaluate surrogate model $\hat{f}$ that emulates the response of expensive computer model f and thus we can use the surrogate model to predict the output of the simulation model for any untried site $\mathbf{x}$. In this section, we describe several popular surrogate modeling techniques.

3.3.1 Polynomial Regression Model (PR)

Polynomial regression (PR) models have been widely used by researchers for modeling computer experiments. PR model uses a polynomial form model to approximate the input-output relationship. The number of terms included in a PR model depends on the desired accuracy of the surrogate and the exact functional form of the relationship between inputs and the response [49, 50]. The general PR model with p basis functions can be expressed as

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{p} \beta_j b_j(\mathbf{x}) = \mathbf{b}^{\mathrm{T}}(\mathbf{x})\boldsymbol{\beta} \tag{3.6}$$

where $\beta_j, j = 1, \cdots, p$ are regression coefficients, $b_j(\mathbf{x}), j = 1, \cdots, p$ are basis functions of the PR model, $\boldsymbol{\beta} = \left[\beta_1, \cdots, \beta_p\right]^{\mathrm{T}}$ is the vector of regression coefficients and $\mathbf{b}(\mathbf{x}) = \left[b_1(\mathbf{x}), \cdots, b_p(\mathbf{x})\right]^{\mathrm{T}}$ is the vector of basis functions. Given the training data set $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \cdots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\}$ defined in Equation (3.5), the PR model in Equation (3.6) for n samples in $\mathbf{X}$ can be written in matrix notion as

$$\hat{\mathbf{y}} = \mathbf{B}\boldsymbol{\beta}, \tag{3.7}$$

where $\hat{\mathbf{y}}$ is the vector of responses given by PR model and $\mathbf{B}$ is the so-called design matrix or Vandermonde matrix, which are respectively defined by Equation (3.8) and (3.9).

$$\hat{\mathbf{y}} = \left[\hat{f}\left(\mathbf{x}^{(1)}\right), \cdots, \hat{f}\left(\mathbf{x}^{(n)}\right) \right]^{\mathrm{T}} = \left[\hat{y}^{(1)}, \cdots, \hat{y}^{(n)} \right]^{\mathrm{T}} \tag{3.8}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}\left(\mathbf{x}^{(1)}\right) \\ \vdots \\ \mathbf{b}\left(\mathbf{x}^{(n)}\right) \end{bmatrix} = \begin{bmatrix} b_1\left(\mathbf{x}^{(1)}\right) & \cdots & b_p\left(\mathbf{x}^{(1)}\right) \\ \vdots & \ddots & \vdots \\ b_1\left(\mathbf{x}^{(n)}\right) & \cdots & b_p\left(\mathbf{x}^{(n)}\right) \end{bmatrix} \tag{3.9}$$

A commonly used approach to estimate the regression coefficients $\boldsymbol{\beta}$ is the least squares method (LSM). The least squares approach is to minimize the sum of squared residuals at the points in the training data set [43], i.e., to minimize

$$S = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 = \sum_{i=1}^{n} \left(y^{(i)} - \sum_{j=1}^{p} \beta_j b_j\left(\mathbf{x}^{(i)}\right) \right)^2 = \left\| \mathbf{y} - \mathbf{B}\boldsymbol{\beta} \right\|^2 . \tag{3.10}$$

If $n \geq p$ and matrix $\mathbf{B}$ is full rank, the least squares estimator of regression coefficients $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{B}^{\mathrm{T}} \mathbf{B} \right)^{-1} \mathbf{B}^{\mathrm{T}} \mathbf{y}. \tag{3.11}$$

Let $\mathbf{x} = \left[x_1, \cdots, x_d \right]^{\mathrm{T}}$ be the d-dimensional design variables, the p polynomial basis functions of first-order (linear) and second-order (quadratic) PR models are:

Linear, $p = d + 1$:

$$b_1\left(\mathbf{x}\right) = 1, \; b_2\left(\mathbf{x}\right) = x_1, \cdots, \; b_{d+1}\left(\mathbf{x}\right) = x_d \tag{3.12}$$

Quadratic, $p = \frac{1}{2}\left(d+1\right)\left(d+2\right)$:

$$\begin{aligned} & b_1\left(\mathbf{x}\right) = 1 \\ & b_2\left(\mathbf{x}\right) = x_1, \cdots, \; b_{d+1}\left(\mathbf{x}\right) = x_d \\ & b_{k+2}\left(\mathbf{x}\right) = x_1^2, \; b_{d+3}\left(\mathbf{x}\right) = x_1 x_2, \cdots, \; b_{2d+1}\left(\mathbf{x}\right) = x_1 x_d \\ & b_{2k+2}\left(\mathbf{x}\right) = x_2^2, b_{d+3}\left(\mathbf{x}\right) = x_2 x_3, \cdots, \; b_{2d+1}\left(\mathbf{x}\right) = x_2 x_d \\ & \cdots \qquad\qquad \cdots \qquad\qquad b_p\left(\mathbf{x}\right) = x_d^2 \end{aligned} \tag{3.13}$$

The number of polynomial basis functions dramatically increases with the number of input variables and the order of the polynomial. Thus, lower-order polynomials such as first-

order and second-order (or quadratic) PR models are commonly used in practice, in order to simplify the modeling process.

With the estimator of regression coefficients $\hat{\boldsymbol{\beta}}$, the approximate response $\hat{f}(\mathbf{x})$ at any untried $\mathbf{x}$ can be predicted by the PR model (3.6). It is worth mentioning that the resulting PR model does not necessarily pass through the training sample data [50], i.e., does not necessarily have a zero error at the training points.

3.3.2 Kriging Model (KG)

Kriging is a Gaussian process [51] based modeling method to interpolate deterministic noise-free data [52] and has proven to be useful in a wide variety of fields [53, 54]. Kriging model is also referred to as "Design and Analysis of Computer Experiments" (DACE) model [46]. The distinctive feature of Kriging model is that it provides not only a predicted response at an unsampled point but also an estimate of the prediction variance. This variance gives indication of the uncertainty in the Kriging model, which results from the construction of the covariance function. The covariance function is based on the idea that when input points are near one another, the correlation between their corresponding outputs will be high. As a result, the uncertainty associated with the model's predictions will be small for input points which are near the training points, and will increase as one moves further from the training points.

The Kriging model consists of two components: a regression function (also known as global trend function) $g(\mathbf{x})$ which is constructed based on the training data set, and a Gaussian process $Z(\mathbf{x})$ which accounts for the local deviation of the data from the trend function. Thus, the general form of Kriging surrogate model is given as

$$\hat{f}(\mathbf{x}) = g(\mathbf{x}) + Z(\mathbf{x}) \tag{3.14}$$

where $g(\mathbf{x})$ is a regression function and $Z(\mathbf{x})$ is assumed to be a realization of Gaussian process with zero mean, variance σ^2 and correlation matrix $\boldsymbol{\Psi}$.

Depending on the form of the regression function, Kriging has been prefixed with different names. Simple Kriging assumes the regression function to be a known constant. A more popular version is ordinary Kriging, which assumes a constant but unknown regression function. In general, universal Kriging treats the trend function as a polynomial regression model, namely,

$$g(\mathbf{x}) = \sum_{j=1}^{p} \beta_j b_j(\mathbf{x}) = \mathbf{b}^{\mathrm{T}}(\mathbf{x})\boldsymbol{\beta} \tag{3.15}$$

where $b_j(\mathbf{x})$ and β_j, $j = 1, \cdots, p$ are basis functions and corresponding coefficients, $\mathbf{b}(\mathbf{x}) = \left[b_1(\mathbf{x}), \cdots, b_p(\mathbf{x})\right]^{\mathrm{T}}$ is the vector of the basis functions, and $\boldsymbol{\beta} = \left[\beta_1, \cdots, \beta_p\right]^{\mathrm{T}}$ is the vector of coefficients.

The idea is that regression function captures the largest variance in the data (the general trend) and that the Gaussian process interpolates the residuals. The covariance matrix of $Z(\mathbf{x})$ is defined as

$$\mathrm{Cov}\left[Z(\mathbf{x}), Z(\mathbf{x}')\right] = \sigma^2 R(\mathbf{x}, \mathbf{x}') \tag{3.16}$$

where $\mathrm{Cov}\left[Z(\mathbf{x}), Z(\mathbf{x}')\right]$ is the covariance of $Z(\mathbf{x})$ between any two points $\mathbf{x}$ and $\mathbf{x}'$, $R(\mathbf{x}, \mathbf{x}')$ is the correlation function between $\mathbf{x}$ and $\mathbf{x}'$, and σ^2 is the stochastic process variance. The correlation function in Equation (3.16) affects the both the range of influence and the smoothness of the model [55]. The most commonly used correlation function is Gaussian correlation function, which is defined by

$$R(\mathbf{x}, \mathbf{x}') = \exp\left[-\sum_{k=1}^{d} \theta_k \left|x_k - x_k'\right|^2\right] \tag{3.17}$$

where θ_k, $j = 1, \cdots, d$ are unknown parameters of the correlation function and can be contained in a column vector $\boldsymbol{\theta} = \left[\theta_1, \cdots, \theta_d\right]^{\mathrm{T}}$, d is the dimension of design space (number of design variables), x_k and x_k' denote the k-th component of the samples $\mathbf{x}$ and $\mathbf{x}'$, respectively, and $\left|x_k - x_k'\right|$ is the absolute distance between x_k and x_k'.

Consider on the training data set $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\} = \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \cdots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\}$, where $\mathbf{X} = \left[\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\right]^{\mathrm{T}}$ are n samples in d-dimensional design space, and $\mathbf{y} = \left[y^{(1)}, \cdots, y^{(n)}\right]^{\mathrm{T}}$ are

the corresponding responses. The prediction mean and prediction variance of Kriging [46] can be respectively given by

$$\mu_{\hat{f}}\left(\mathbf{x}\right)=\mathbf{b}^{\mathrm{T}}\left(\mathbf{x}\right)\boldsymbol{\beta}+\mathbf{r}^{\mathrm{T}}\left(\mathbf{x}\right)\mathbf{R}^{-1}\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right) \tag{3.18}$$

$$s_{\hat{f}}^{2}\left(\mathbf{x}\right)=\sigma^{2}\left(1-\begin{bmatrix}\mathbf{b}^{\mathrm{T}}\left(\mathbf{x}\right) & \mathbf{r}^{\mathrm{T}}\left(\mathbf{x}\right)\end{bmatrix}\begin{bmatrix}0 & \mathbf{F}^{\mathrm{T}}\\ \mathbf{F} & \mathbf{R}\end{bmatrix}\begin{bmatrix}\mathbf{b}^{\mathrm{T}}\left(\mathbf{x}\right)\\ \mathbf{r}^{\mathrm{T}}\left(\mathbf{x}\right)\end{bmatrix}\right) \tag{3.19}$$

where $\mathbf{b}\left(\mathbf{x}\right)=\left[b_1\left(\mathbf{x}\right),\cdots,b_p\left(\mathbf{x}\right)\right]^{\mathrm{T}}$ is the regression basis function vector of the predicting point $\mathbf{x}$, $\boldsymbol{\beta}=\left[\beta_1,\cdots,\beta_p\right]^{\mathrm{T}}$ is the regression coefficients, $\mathbf{r}\left(\mathbf{x}\right)$ is the vector of correlation functions between the untried point and the n sampled (observed) points (defined in Equation (3.20)), $\mathbf{R}$ is the $n\times n$ matrix of correlation functions for the sampled data (defined by Equation (3.21)), and $\mathbf{F}$ is the $n\times p$ model matrix defined by (3.22).

$$\mathbf{r}\left(\mathbf{x}\right)=\begin{bmatrix}R\left(\mathbf{x},\mathbf{x}^{(1)}\right) & R\left(\mathbf{x},\mathbf{x}^{(2)}\right) & \cdots & R\left(\mathbf{x},\mathbf{x}^{(n)}\right)\end{bmatrix}^{\mathrm{T}} \tag{3.20}$$

$$\mathbf{R}=\begin{bmatrix}R\left(\mathbf{x}^{(1)},\mathbf{x}^{(1)}\right) & R\left(\mathbf{x}^{(1)},\mathbf{x}^{(2)}\right) & \cdots & R\left(\mathbf{x}^{(1)},\mathbf{x}^{(n)}\right)\\ R\left(\mathbf{x}^{(2)},\mathbf{x}^{(1)}\right) & R\left(\mathbf{x}^{(2)},\mathbf{x}^{(2)}\right) & \cdots & R\left(\mathbf{x}^{(2)},\mathbf{x}^{(n)}\right)\\ \vdots & \vdots & \ddots & \vdots\\ R\left(\mathbf{x}^{(n)},\mathbf{x}^{(1)}\right) & R\left(\mathbf{x}^{(n)},\mathbf{x}^{(2)}\right) & \cdots & R\left(\mathbf{x}^{(n)},\mathbf{x}^{(n)}\right)\end{bmatrix} \tag{3.21}$$

$$\mathbf{F}=\begin{bmatrix}b_1\left(\mathbf{x}^{(1)}\right) & \cdots & b_p\left(\mathbf{x}^{(1)}\right)\\ \vdots & \vdots & \vdots\\ b_1\left(\mathbf{x}^{(n)}\right) & \cdots & b_p\left(\mathbf{x}^{(n)}\right)\end{bmatrix} \tag{3.22}$$

From above description, in order to build Kriging surrogate model, the regression coefficients $\boldsymbol{\beta}$ in Equation (3.15), the correlation parameter $\boldsymbol{\theta}$ in Equation (3.17), and the stochastic process variance σ^2 in Equation (3.16) need to be determined. The maximum likelihood estimate method is used to determine the unknown model parameters $\boldsymbol{\beta}$, σ^2 and $\boldsymbol{\theta}$.

Since the Kriging method assumes that the observed responses are from a Gaussian process, the responses at sampling sites are considered to be correlated random functions with the corresponding likelihood function given by

$$L\left(\boldsymbol{\beta},\sigma^2,\boldsymbol{\theta}\right)=\frac{1}{\left(2\pi\sigma^2\right)^{n/2}\left|\mathbf{R}\right|^{1/2}}\exp\left[-\frac{\left(\mathbf{y}_{\mathrm{S}}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right)^{\mathrm{T}}\mathbf{R}^{-1}\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right)}{2\sigma^2}\right] \tag{3.23}$$

To simplify above likelihood function, we take the natural logarithm and gives

$$\ln\left(L\right)=-\frac{n}{2}\ln\left(2\pi\right)-\frac{n}{2}\ln\left(\sigma^2\right)-\frac{1}{2}\left|\mathbf{R}\right|-\frac{\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right)^{\mathrm{T}}\mathbf{R}^{-1}\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right)}{2\sigma^2} \tag{3.24}$$

By taking derivatives of Equation (3.24) with respect to $\boldsymbol{\beta}$ and σ^2 respectively, and setting to zero, we obtain maximum likelihood estimates (MLEs) for $\boldsymbol{\beta}$ and σ^2

$$\hat{\boldsymbol{\beta}}=\left(\mathbf{F}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{F}\right)^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y} \tag{3.25}$$

$$\hat{\sigma}^2=\frac{1}{n}\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right)^{\mathrm{T}}\mathbf{R}^{-1}\left(\mathbf{y}-\mathbf{F}^{\mathrm{T}}\boldsymbol{\beta}\right) \tag{3.26}$$

By substituting Equation (3.25) and (3.26) into Equation (3.24), we obtain the ln-likelihood function only in terms of parameter vector $\boldsymbol{\theta}$, which is known as the concentrated ln-likelihood function:

$$L\left(\boldsymbol{\theta}\right)=\ln\left(L\right)=-\frac{n}{2}\ln\left(2\pi\right)-\frac{n}{2}\ln\left(\hat{\sigma}^2\right)-\frac{1}{2}\ln\left|\mathbf{R}\right| \tag{3.27}$$

The estimator $\hat{\boldsymbol{\theta}}$ of parameter vector $\boldsymbol{\theta}$ is obtained by maximize Equation (3.27) under the constraint $\theta_l>0,\ l=1,2,\cdots,d$. In other words, parameter vector $\boldsymbol{\theta}$ is achieved by solving the following optimization problem:

$$\begin{cases}\underset{\boldsymbol{\theta}}{\mathrm{Max}}\ L\left(\boldsymbol{\theta}\right)\\ \text{s.t. }\ \theta_l>0,\ l=1,2,\cdots,k\end{cases} \tag{3.28}$$

This optimization problem can be carried out using numerical optimization technique. A global search method such as a genetic algorithm or simulated annealing usually produce good result. After getting $\hat{\boldsymbol{\theta}}$, the estimators $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ can be calculated by Equation (3.25) and (3.26). So far, we have determined the values of all unknowns in Kriging model, i.e., the Kriging model has been completely built. The prediction at any untried point can be estimated by Equation (3.18).

3.3.3 Radial Basis Function Model (RBF)

Radial basis function (RBF) model, which is also called Radial basis function network, can be interpreted as an artificial neural network that uses radial basis functions as activation functions, as shown in Figure 3.4. A RBF network consists of three layers: an input layer, a hidden layer with a non-linear RBF activation function, and a linear output layer [56]. Usually, the neurons (nodes) in the hidden layer use the RBF in form of $\phi_i(\mathbf{x}) = \phi\left(\left\|\mathbf{x}-\mathbf{c}^{(i)}\right\|\right)$, $i = 1,\cdots,m$, where $\mathbf{x}$ is the input, m is the number of neurons in hidden layer, $\mathbf{c}^{(i)}$ is the center or prototype of the *i*-th radial basis function $\phi_i(\mathbf{x})$, and $\|\cdot\|$ is usually a Euclidean norm. For input $\mathbf{x} = [x_1,\cdots,x_d]^{\mathrm{T}}$, the output of the RBF network $\hat{f}(\mathbf{x})$ is given by a linear combination of a set of radial basis functions

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi\left(\left\|\mathbf{x}-\mathbf{c}^{(i)}\right\|\right) \tag{3.29}$$

where w_i is the weight for *i*-th radial basis function. The RBFs $\phi\left(\left\|\mathbf{x}-\mathbf{c}^{(i)}\right\|\right)$, $i = 1,\cdots,m$ transform the input in terms of the distance between the input $\mathbf{x}$ and the center $\mathbf{c}^{(i)}$. For the generalized RBF network, the center $\mathbf{c}^{(i)}$ are also unknown and have to be learned by other methods such as the *k*-means method [2].

Commonly used non-parametric basis functions are [50] (writing $r = \|\mathbf{x}-\mathbf{c}\|$):

- Linear: $\phi(r) = r$,
- Cubic: $\phi(r) = r^3$, and
- Thin plate spline: $\phi(r) = r^2 \ln(r)$.

To improve the generalization properties of RBF model (3.29), parametric basis functions can be adopted. Commonly used types of parametric basis functions include:

- Gaussian: $\phi(r) = e^{-r^2/(2\sigma^2)}$,
- Multi-quadric: $\phi(r) = \left(r^2+\sigma^2\right)^{1/2}$, and
- Inverse multi-quadric: $\phi(r) = \left(r^2+\sigma^2\right)^{-1/2}$.

Whether we choose a set of parametric basis function or non-parametric ones, the weights $\mathbf{w}=\left[w_1,\cdots,w_m\right]^{\mathrm{T}}$ can be evaluated via the interpolation condition [54]. With the training points $\mathbf{X}=\left[\mathbf{x}^{(1)},\cdots,\mathbf{x}^{(n)}\right]^{\mathrm{T}}$ and their corresponding responses $\mathbf{y}=\left[y^{(1)},\cdots,y^{(n)}\right]^{\mathrm{T}}$, the interpolation condition [57] can be written in matrix form as

$$\mathbf{\Phi w}=\mathbf{y} \tag{3.30}$$

where

$$\mathbf{\Phi}=\begin{bmatrix}\phi\left(\left\|\mathbf{x}^{(1)}-\mathbf{c}^{(1)}\right\|\right) & \phi\left(\left\|\mathbf{x}^{(1)}-\mathbf{c}^{(2)}\right\|\right) & \cdots & \phi\left(\left\|\mathbf{x}^{(1)}-\mathbf{c}^{(m)}\right\|\right)\\ \phi\left(\left\|\mathbf{x}^{(2)}-\mathbf{c}^{(1)}\right\|\right) & \phi\left(\left\|\mathbf{x}^{(2)}-\mathbf{c}^{(2)}\right\|\right) & \cdots & \phi\left(\left\|\mathbf{x}^{(2)}-\mathbf{c}^{(m)}\right\|\right)\\ \vdots & \vdots & \ddots & \vdots\\ \phi\left(\left\|\mathbf{x}^{(n)}-\mathbf{c}^{(1)}\right\|\right) & \phi\left(\left\|\mathbf{x}^{(n)}-\mathbf{c}^{(2)}\right\|\right) & \cdots & \phi\left(\left\|\mathbf{x}^{(n)}-\mathbf{c}^{(m)}\right\|\right)\end{bmatrix} \tag{3.31}$$

It is easy to see that if $m=n$ and the centers of the radial basis functions coincide with the training points $\mathbf{c}^{(i)}=\mathbf{x}^{(i)},i=1,\cdots,n$, $\mathbf{\Phi}$ is a regular square matrix and the matrix equation (3.30) has unique solution $\mathbf{w}=\mathbf{\Phi}^{-1}\mathbf{y}$.

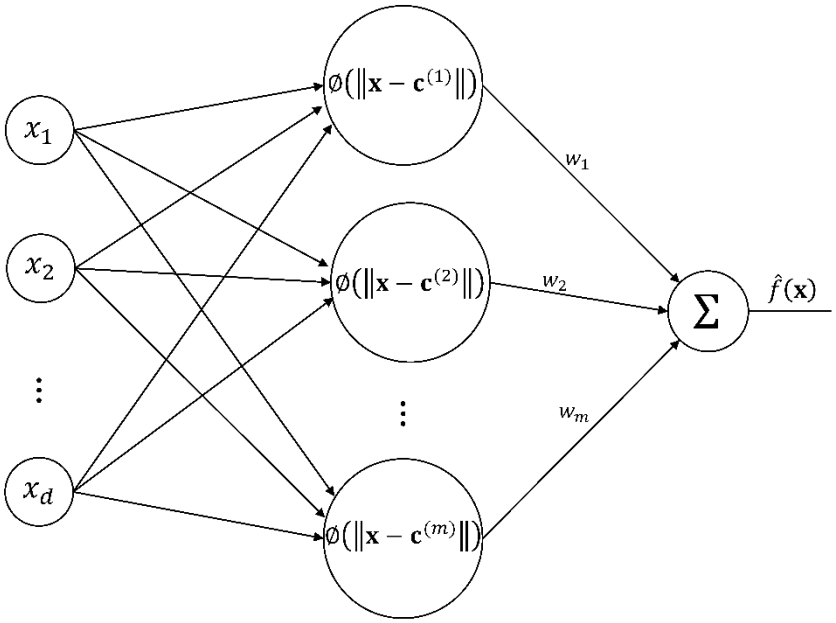

Figure 3.4 Structure of RBF network models

It is important to keep in mind that RBFs are essentially interpolating functions (i.e., a trained RBF model will pass through all the training points). This property enables RBFs to

represent highly nonlinear data, which is otherwise often challenging to accomplish using low order PR models [50].

3.3.4 Multi-layer Perceptron Networks (MLP)

Multi-layer perceptron (MLP) network, which is a feedforward artificial neural network (ANN) model that maps sets of input data onto a set of appropriate outputs, is a more recent and increasingly popular choice of surrogate model. A MLP network consists of an input layer, several hidden layers, and an output layer. The neuron or node which includes a summation and activation function is the basis structure of ANN.

A simple MLP network (shown in Figure 3.5), that consists of one input layer, one hidden layer with nonlinear activation function, and one output layer with linear activation function, approximates the inputs and outputs as follows:

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{m} \beta_j g\left(\sum_{i=1}^{d} w_{ij} x_i + w_{0j} \right) + \beta_0 \tag{3.32}$$

where $\mathbf{x} = [x_1, \cdots, x_d]^{\mathrm{T}}$ is the input, m is the number of nodes (neurons) of the hidden layer, β_j is the weight connect between the output and the *j*-th note in the hidden layer, the function $g(\cdot)$ is the activation function of the hidden layer, w_{ij} is the weight connection between the *i*-th component of the input and the *j*-th note in the hidden layer. The most commonly used activation function is the logistic function (or called Sigmoid function), which has the form:

$$g(net) = \frac{1}{1 + \exp(-\lambda \cdot net)} \tag{3.33}$$

where λ is a constant and $net = \sum_{i=1}^{d} w_i x_i + w_0$ is the so-called net input signal to the neuron. A comprehensive study can be found in [58, 59].

Then, training a network is to estimate the unknown parameters in (3.32). Usually, MLP utilizes a supervised learning technique called backpropagation for training the network [2, 59]. One of the drawbacks associated with neural networks for function approximation is the fairly large number of parameters that need to be prescribed by the user, thereby demanding adequate user experience in implementing MLP. These prescribed

parameters include the number of neurons, the number of layers, the type of activation function, and the optimization algorithm used to train the network. In addition, the training process generally needs to be supervised in order to avoid "over-fitting" [50].

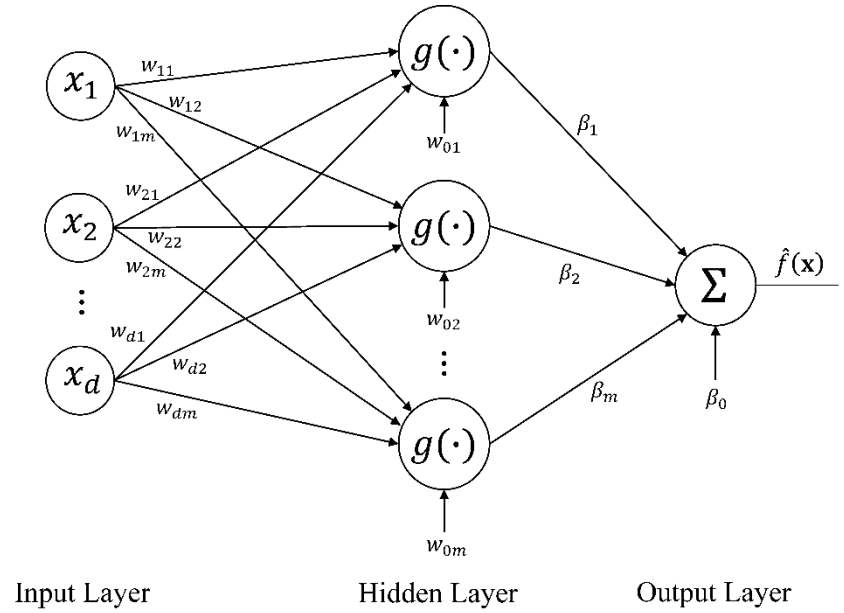

Figure 3.5 Structure of a three-layer MLP network

3.3.5 Support Vector Regression (SVR)

Support vector machines (SVM) are popular models for solving classification and regression problems in recent years. An important and favorable property of SVM models is that the determination of the model parameters corresponds to a convex optimization problem [60], i.e., there is no local minima during training SVM models, and thus the optimization process does not depend on the problem dimensions and overfitting is seldom an issue [18, 20]. In support vector regression (SVR), the goal is to construct a model $\hat{f}(\mathbf{x})$ that has at most an ε deviation from the actual targets $y^{(i)}$ for all the training data, and at the same time is as flat as possible [61]. In other words, the errors are not considered if they are less than ε, but any deviation larger than this will not be accepted.

Given the training data set $\mathcal{D}=\{\mathbf{X},\mathbf{y}\}=\left\{\left(\mathbf{x}^{(1)},y^{(1)}\right),\cdots,\left(\mathbf{x}^{(n)},y^{(n)}\right)\right\}$ where $\mathbf{x}\in\mathbb{R}^d$ and $y\in\mathbb{R}$, a general SVR model can be given as

$$\hat{f}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b \tag{3.34}$$

where $\phi(\mathbf{x})$ is a map (or transform) that transforms the input space $\mathcal{X}$ (when $\mathbf{x} \in \mathbb{R}^d$, $\mathcal{X}$ is $\mathbb{R}^d$) to some feature space $\mathcal{H}$, i.e., map $\phi(\mathbf{x}): \mathcal{X} \mapsto \mathcal{H}$, $\mathbf{w} \in \mathcal{H}$ is the column vector of weights, $b \in \mathbb{R}$ is the bias. A small $\mathbf{w}$ in Equation (3.34) means that the regression is flat [61]. One way to ensure the flatness is to minimize the norm, $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ where $\langle \cdot, \cdot \rangle$ denotes the dot product. Considering minimizing this norm under the restriction of deviation, the construction of the SVR model is reduced to the following convex optimization problem:

$$\begin{aligned} &\text{minmize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} \begin{cases} y^{(i)} - \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right) - b \le \varepsilon \\ \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right) + b - y^{(i)} \le \varepsilon \end{cases} \end{aligned} \tag{3.35}$$

Note that the tacit assumption in Equation (3.35) is that a function $\hat{f}(\mathbf{x})$ exists that approximates all pairs $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$ in training data set with precision ε, or in other words, that the convex optimization problem in (3.35) is feasible [57]. However, sometimes such a solution may not actually exist and it is also likely that better predictions will be obtained if we allow for the possibility of outliers. This is achieved by introducing slack variables $\xi^{*(i)} \ge 0$ for $\mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right) + b - y^{(i)} \ge \varepsilon$ and $\xi^{(j)} \ge 0$ for $y^{(j)} - \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(j)}\right) - b \ge \varepsilon$. Thus, the optimization problem in (3.35) can be rewritten as:

$$\begin{aligned} &\text{minmize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\left(\xi^{(i)} + \xi^{*(i)}\right) \\ &\text{subject to} \begin{cases} y^{(i)} - \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right) - b \le \varepsilon + \xi^{(i)} \\ \mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right) + b - y^{(i)} \le \varepsilon + \xi^{*(i)} \\ \xi^{(i)}, \xi^{*(i)} \ge 0, \qquad i = 1, \cdots, n \end{cases} \end{aligned} \tag{3.36}$$

The minimization in Equation (3.36) is a trade-off between the flatness of $\hat{f}(\mathbf{x})$ (model complexity) and the degree to which deviations larger than ε are tolerated. This trade-off is controlled by the user defined constant $C > 0$. This method of tolerating errors is known as ε-insensitive loss function [62]. The linear ε-insensitive loss function is described by Equation (3.37),, which is shown in the right part of Figure 3.6. Points that lie inside the ε-

insensitive tube (the region between two red dashed lines in the left part of Figure 3.6) have no loss associated with them, only points outside contribute to the loss of the function.

$$L_\varepsilon\left(y(\mathbf{x}),\hat{f}(\mathbf{x})\right)=\begin{cases}0, & \left|\hat{f}(\mathbf{x})-y(\mathbf{x})\right|\le\varepsilon\\ \left|\hat{f}(\mathbf{x})-y(\mathbf{x})\right|-\varepsilon, & \text{otherwise}\end{cases} \tag{3.37}$$

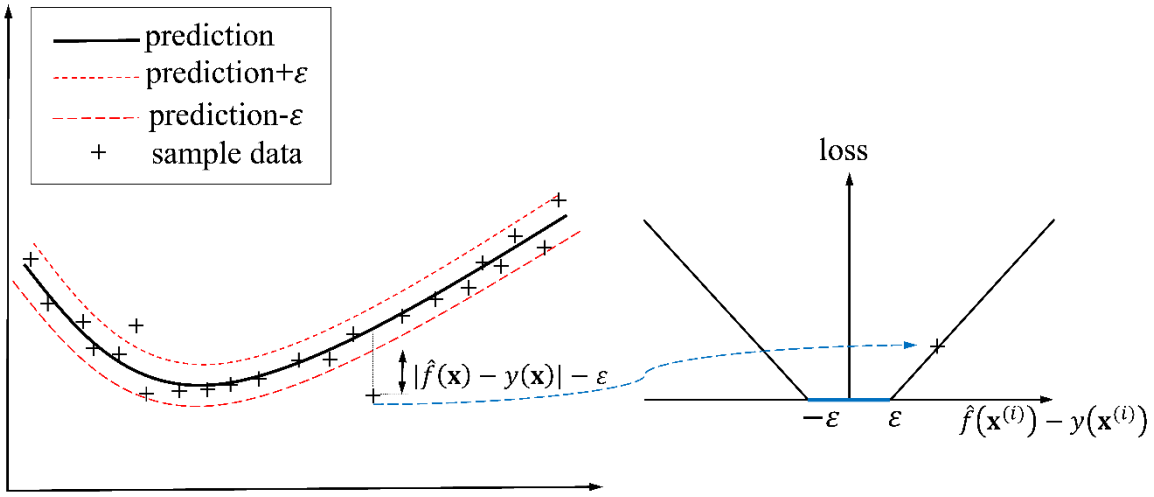

Figure 3.6 The ε-insensitive loss function for SVR

The constrained optimization problem of Equation (3.36) can be solved more easily in its dual formulation based on Lagrange multipliers. By introducing Lagrange multipliers, $\mu^{(i)}$, $\mu^{*(i)}$, $\alpha^{(i)}$ and $\alpha^{*(i)}$, the Lagrange function (or Lagrangian) L is given as

$$\begin{aligned}L=&\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\left(\xi^{(i)}+\xi^{*(i)}\right)-\sum_{i=1}^{n}\left(\mu^{(i)}\xi^{(i)}+\mu^{*(i)}\xi^{*(i)}\right)\\&-\sum_{i=1}^{n}\left[\alpha^{(i)}\left(\varepsilon+\xi^{(i)}-y^{(i)}+\mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right)+b\right)\right]\\&-\sum_{i=1}^{n}\left[\alpha^{*(i)}\left(\varepsilon+\xi^{*(i)}+y^{(i)}-\mathbf{w}^{\mathrm{T}}\phi\left(\mathbf{x}^{(i)}\right)-b\right)\right]\end{aligned} \tag{3.38}$$

The Lagrangian L must be minimized with respect to $\mathbf{w}$, b, $\xi^{(i)}$ and $\xi^{*(i)}$ (the primal variables) and maximized with respect to $\mu^{(i)}$, $\mu^{*(i)}$, $\alpha^{(i)}$ and $\alpha^{*(i)}$ (the dual variables), where $\mu^{(i)},\mu^{*(i)},\alpha^{(i)},\alpha^{*(i)}\ge 0$. Then, the minimization of L with respect to the primal variables ($\mathbf{w}$, b, $\xi^{(i)}$, $\xi^{*(i)}$) and the maximization with respect to dual variables involves finding the saddle point, at which the derivatives of L with respect to the primal variables have to vanish, i.e.,

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\left[\left(\alpha^{(i)} - \alpha^{*(i)}\right)\phi\left(\mathbf{x}^{(i)}\right)\right] = 0 \tag{3.39}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\left(\alpha^{(i)} - \alpha^{*(i)}\right) = 0 \tag{3.40}$$

$$\frac{\partial L}{\partial \xi^{(i)}} = C - \alpha^{(i)} - \mu^{(i)} = 0 \tag{3.41}$$

$$\frac{\partial L}{\partial \xi^{*(i)}} = C - \alpha^{*(i)} - \mu^{*(i)} = 0 \tag{3.42}$$

Substituting Equations (3.39), (3.40), (3.41) and (3.42) into Equation (3.38) to eliminate $\mu^{(i)}$ and $\mu^{*(i)}$, finally yields the dual optimization problem:

$$\begin{aligned} &\text{maximize} \quad \begin{cases} L'\left(\alpha,\alpha^*\right) = -\dfrac{1}{2}\displaystyle\sum_{i,j=1}^{n}\left(\alpha^{(i)} - \alpha^{*(i)}\right)\left(\alpha^{(j)} - \alpha^{*(j)}\right)k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right) \\ \qquad\qquad -\varepsilon\displaystyle\sum_{i=1}^{n}\left(\alpha^{(i)} + \alpha^{*(i)}\right) + \sum_{i=1}^{n} y^{(i)}\left(\alpha^{(i)} - \alpha^{*(i)}\right) \end{cases} \\ &\text{subject to} \quad \sum_{i=1}^{n}\left(\alpha^{(i)} - \alpha^{*(i)}\right) = 0 \quad \text{and} \quad \alpha^{(i)}, \alpha^{*(i)} \in [0, C] \end{aligned} \tag{3.43}$$

where $k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right) = \left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right)\right\rangle = \left[\phi\left(\mathbf{x}^{(i)}\right)\right]^{\mathrm{T}} \phi\left(\mathbf{x}^{(j)}\right)$ is the kernel function. This maximize problem usually solved by quadratic programming algorithms, and thus $\alpha^{(i)}$ and $\alpha^{*(i)}$ are obtained.

From Equation (3.39) we can obtain

$$\mathbf{w} = \sum_{i=1}^{n}\left[\left(\alpha^{(i)} - \alpha^{*(i)}\right)\phi\left(\mathbf{x}^{(i)}\right)\right] \tag{3.44}$$

Thus, the prediction of SVR for any new input point is computing by

$$\hat{f}\left(\mathbf{x}\right) = \sum_{i=1}^{n}\left(\alpha^{(i)} - \alpha^{*(i)}\right)k\left(\mathbf{x},\mathbf{x}^{(i)}\right) + b \tag{3.45}$$

where $k\left(\mathbf{x},\mathbf{x}^{(i)}\right) = \left\langle \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}^{(i)}\right)\right\rangle = \left[\phi\left(\mathbf{x}\right)\right]^{\mathrm{T}} \phi\left(\mathbf{x}^{(i)}\right)$ is the kernel function between the new input $\mathbf{x}$ and sampled point $\mathbf{x}^{(i)}$ in training data set. The constant term b, known as the bias, is obtained by exploiting the so-called Karush-Kuhn-Tucker (KKT) conditions[61, 63].

The use of kernel functions (or kernels) $k(\cdot,\cdot)$ make SVR models have the capability of capturing complicated landscape. The kernel is related to the transformation $\phi\left(\mathbf{x}\right)$, which maps the input space $\mathcal{X}$ to the so-called feature space $\mathcal{H}$ (i.e. $\phi\left(\mathbf{x}\right): \mathcal{X} \mapsto \mathcal{H}$), by

$k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)=\left\langle\phi\left(\mathbf{x}^{(i)}\right),\phi\left(\mathbf{x}^{(j)}\right)\right\rangle$. In other words, the kernels are the inner products (also known as dot products) of input points. Conventionally, kernels need to satisfies continuous, symmetric and positive definite conditions. Popular choices for $k\left(\cdot,\cdot\right)$ are:

- Linear: $k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)=\left\langle\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right\rangle$
- p degree homogeneous polynomial: $k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)=\left(\left\langle\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right\rangle\right)^{p}$
- p degree inhomogeneous polynomial: $k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)=\left(\left\langle\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right\rangle+c\right)^{p}$
- Gaussian: $k\left(\mathbf{x}^{(i)},\mathbf{x}^{(j)}\right)=\exp\left(-\left|\mathbf{x}^{(i)}-\mathbf{x}^{(j)}\right|^{2}\Big/\sigma^{2}\right)$

3.4 Model Validation/Model Performance Assessment

The accuracy of a surrogate model is affected by the type of surrogate model and the quality and quantity of the dataset from which it is constructed [64]. Before we use the built surrogate models, the performance of the surrogate model need to be assessed and validated. This section discusses several measures and methods to assess the accuracy of a surrogate model.

3.4.1 Fitting Error and Prediction Error

The most popular and simple way to evaluate the accuracy of a surrogate model is to examine its residual errors, i.e., the difference between the output from original simulation model and that predicted by the surrogate model. Let $\mathbf{x}=\left[x_1,\cdots,x_d\right]^{\mathrm{T}}$ denotes the vector of input variables, y denotes the output from simulation model $f\left(\mathbf{x}\right)$, and $\hat{y}$ signifies the output predicted by surrogate model $\hat{f}\left(\mathbf{x}\right)$ built on training data set $\mathcal{D}=\left\{\left(\mathbf{x}^{(1)},y^{(1)}\right),\cdots,\left(\mathbf{x}^{(n)},y^{(n)}\right)\right\}$. The residual error thus can be expressed as $\left|f\left(\mathbf{x}\right)-\hat{f}\left(\mathbf{x}\right)\right|$. Residual error evaluated on the training points is referred to as the fitting error, while residual error at the set of points which are randomly generated and not be touched during the surrogate model constructing stages is called prediction error. In the following, several popular measures of model accuracy that are based on residual error and thus can be evaluated on whether fitting error or prediction error are presented.

The first two measures are the root mean squared error (RMSE) and the maximum absolute error (MAE) [65], separately defined by

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2} \tag{3.46}$$

$$MAE = \max\left|y^{(i)} - \hat{y}^{(i)}\right|, \quad i = 1, 2, \cdots, m \tag{3.47}$$

where m is the number of validation points, $y^{(i)}$ is output from experiment, and $\hat{y}^{(i)}$ is the output predicted by surrogate model. When RMSE and MAE are calculated on fitting error, the training points are used as validation points. While, a set of untried points are used when we assess the surrogate model by prediction error.

An additional error measure based on the residual is the coefficient of determination R^2, which is defined by

$$R^2 = 1 - \frac{\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\sum_{i=1}^{m}\left(y^{(i)} - \bar{y}\right)^2} = \frac{\sum_{i=1}^{m}\left(\hat{y}^{(i)} - \bar{y}\right)^2}{\sum_{i=1}^{m}\left(y^{(i)} - \bar{y}\right)^2} \tag{3.48}$$

where m is the number of training pints (sampling points), $y^{(i)}$, $\hat{y}^{(i)}$ and $\bar{y}$ respectively represent the (or experimental) response, the prediction and the mean of responses. The value of coefficient of determination R^2 is between 0 and 1, i.e., $0 \le R^2 \le 1$. If R^2 is near to 1, generally, it indicates the model fit well the sampled data.

For surrogate models that interpolate the training points, there are no residual error on training points, i.e., the fitting error is zero. Consequently, measures based on fitting error are not relevant for interpolating surrogate models. In this case, the prediction error or the cross validation described in next subsection can be used to assess the performance of the model. Usually, prediction error, which assesses the ability of the surrogate model to predict responses in unknown design points, is a more useful measure than fitting error.

3.4.2 Cross Validation

In many situations, computer experiments are computationally expensive. Thus, in such situations, evaluation of the prediction error is expensive or time-consuming. To alleviate

this problem, a general strategy is to estimate the prediction error of a surrogate model using the cross validation (CV) procedure [43].

In cross validation (CV), the sample data is divided into training and test points. The training points are used to build the surrogate model, while the test points are used to test the performance of the model. The cross-validation technique operates through the following five steps [50]:

1) Splits the sample points randomly into p (approximately) equal subsets;
2) Removes each of these subsets in turn (one at a time);
3) Trains a surrogate model according to the remaining $p-1$ subsets;
4) Computes the error of the built surrogate model using the omitted subset;
5) Once each one of the p subsets has been used as the omitted subset, the p sets of errors are generally aggregated to yield a global error measure.

Above described CV technique, which splits the sampling data into p subsets, is known as p-fold CV. A variation of p-fold CV is the leave-k-out approach, in which all possible subsets of size k are left out, and the surrogate model is constructed to the remaining set. Each time, the error measure is evaluated at the omitted points. If $k=1$, the cross validation in this special case is called leave-one-out CV. Leave-one-out CV is probably the simplest and most widely used method for surrogate model validation when additional validation points are not available. The root mean squares prediction error for the one-leave-out CV is calculated by:

$$RMSE_{CV}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}-\hat{y}_{-i}^{(i)}\right)^2} \tag{3.49}$$

where $y^{(i)}$ is the response at $\mathbf{x}^{(i)}$ from sample data and $\hat{y}_{-i}^{(i)}$ is the prediction at $\mathbf{x}^{(i)}$ from the surrogate model constructed by using the sample points except $\left(\mathbf{x}^{(i)},y^{(i)}\right)$. The leave-one-out CV is a measure of how sensitive the surrogate model is to lost information at its data points. However, an insensitive surrogate model is not necessarily accurate and an accurate model is not necessarily insensitive to lost information. Therefore, the leave-one-out CV is not

sufficient to measure surrogate model accuracy, and the validation with an additional data set is hence recommended [66].

CV is an extremely popular method for verifying the prediction capability of a surrogate model, when additional validation points are unavailable or expensive. The disadvantage of this method is that the surrogate model has to be constructed more than once. For example, in a leave-one-out CV with n samples, the surrogate model need to be trained for n times. If the construction of the surrogate model is expensive, CV would be impractical.

4. Kriging-Assisted CMA Evolution Strategy

This chapter focuses on Kriging-assisted evolution strategies, under the framework of surrogate-assisted evolutionary optimization. A brief introduction of surrogate-assisted evolution strategy is presented at Section 4.1. Then, the incorporation mechanisms of surrogate model into ES and challenges exist in this domain are described in Section 4.2, based on reviewing of literature. In Section 4.3, new methods about training set selection, pre-selection, evolution control have been developed. Correspondingly, concrete algorithms of Kriging-assisted CMA-ES algorithms are detailed in Section 4.4. In Section 4.5, experimental studies of Kriging-assisted CMA-ES are performed to investigate and analyze the performance of Kriging-assisted CMA-ES algorithms. Finally, a summary of this chapter is provided in Section 4.6.

4.1 Introduction

For continuous optimization problems, evolution strategy (ES) generally works better than other evolutionary algorithms, such as genetic algorithm (GA) [67]. In recent black-box optimization benchmarking (2009 and 2010 GECCO), CMA-ES shows its outstanding performance and has proven to be one of the best-performing search strategies for real-valued black-box optimization [68]. Therefore, in this thesis, we adopt ES to solve expensive optimization problems. However, ES, like other population based EAs, require a large number of fitness function evaluations before obtaining a satisfying solution. Additionally, in expensive optimization problems of real world applications, such as engineering design optimization, no analytical fitness function exists and the evaluations of fitness function are by means of expensive numerical simulations or experiments. Thus, large number of fitness evaluations is not practical or prohibitive. Computational cost, consequently, has been a crucial challenge in application of ES (and other EAs) to expensive optimization problems.

This challenge has stimulated the advent and development of surrogate-assisted ES (also called fitness approximation in ES), which becomes a promising solution to reduce the computation cost of ES. In surrogate-assisted evolution strategies, generally, with the evaluated points, a surrogate model is trained to approximate the real (or original) fitness function and then used together with the real fitness function to guide the search of promising solutions. In this way, the computational cost is reduced because the evaluation of a surrogate is much cheaper than that of the expensive fitness function.

Among the ESs, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a highly developed evolution strategy and has become a standard for continuous black-box evolutionary optimization. It is a powerful optimization algorithm and performs especially well in non-smooth, multimodal back-box problems. The CMA-ES adopts the correlated mutation operator, which makes it a high-level algorithm compared with other algorithms that use isotropic mutation. In CMA-ES, two techniques, namely the covariance matrix adaptation (CMA) and the cumulative step-size adaptation (CSA), are used for adapting the covariance matrix of mutation and the step-size, respectively. The CMA-ES is selected as the basis evolution strategy for surrogate-assisted ES in this work owing to its powerfulness and success in continuous black-box optimization.

Compared with other surrogate models like RBF, MLP model, the advantage of Kriging model is that it not only predicts fitness value but also provides the standard deviation for the predicted fitness without additional computational cost. The standard deviation of predicted fitness indicates the uncertainty in fitness approximation. This information is valuable for evolution control and improving the quality of surrogate in evolutionary computation process. That is the reason for selecting Kriging model for surrogate-assisted evolution strategy. Consequently, this chapter concentrates Kriging-Assisted Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is abbreviated as KA-CMA-ES in the thesis.

4.2 A Brief Survey of Surrogate-Assisted Evolution Strategies

The essential goal of surrogate-assisted evolutionary optimization is to use one or several surrogate models $\hat{f}$ of the fitness function (objective function) f to improve the quality of the search especially in terms of number of functions evaluations required to reach the optimum. It is usually assumed that the evaluation of f is expensive in terms of time or money and the learning (or training) of $\hat{f}$ is relatively cheap. Surrogate model $\hat{f}$ is trained (learned/built) from a set of pairs $\left(\mathbf{x}, f\left(\mathbf{x}\right)\right)$ which is known as training data set or simply training set. Thus, the selection of training set has influence on the quality of the surrogate model developed thereof. Generally speaking, a surrogate model with good quality ensures good performance in surrogate-assisted optimization. In this section, a short overview of existed works on surrogate-assisted evolution strategies (ES) is presented and some challenges in this field are described subsequently.

4.2.1 Surrogate-Assisted Evolution Strategies

The first surrogate-assisted ES, very likely, were proposed by M. Papadrakakis in [69], where a hidden-layer ANN was used to predict the objective and constraint functions values of an expensive structural optimization problem using $\left(\mu,\lambda\right)$-ES and $\left(\mu+\lambda\right)$-ES . This hybrid optimization procedure based on the combination of ES and ANN was found to be very effective in shape and sizing structural optimization problems.

Individual-based and generation-based evolution control methods were proposed in [70] for surrogate-assisted evolution strategies with CMA-ES. Evolution control is popular in surrogate-assisted evolutionary algorithms, including evolution strategies. In individual-based control, part of the individuals in the population are chosen and evaluated with the original fitness function. The controlled individuals can be chosen by random or best strategies. In generation-based control, the whole population of η generations will be evaluated with original fitness function in every p generations, where $\eta \le p$. Reference [70] focused on generation-based evolution control and proposed a strategy to determine the control frequency η / p . Specifically, the current model error was used to estimate the local

fidelity of the approximate model and the local model fidelity was adopted to determine the control frequency. After each generation control, new data points were added to training data set and then the approximate model was updated. In the on-line learning of approximate model, the weighted learning using covariance matrix was proposed and used in this work. In [71], generation-based evolution control was adopted in surrogate-assisted CMA-ES, where Gaussian process and random forests models were used as surrogates.

Pre-selection (pre-screening) of promising solutions (based on approximate model) is another popular strategy of exploiting information from the surrogate model $\hat{f}$ in Evolution Strategies. In pre-selection strategy, $\lambda_{\text{Pre}} > \lambda$ (λ is the population size of ES) individuals are generated each generation, then all λ_{Pre} individuals are evaluated by approximate model $\hat{f}$, after that, λ out of λ_{Pre} best individuals are selected to evaluate by the original fitness function. The basic idea behind this approach is that only the most promising individuals with a good fitness prediction are evaluated with the true fitness function, which results in a reduction of the number of expensive true fitness calls. The Model-Assisted Evolution Strategy (MAES) using pre-selection strategy for (μ, λ)-ES was described in [72].

Applications of Kriging model based pre-selection strategy in ES were studied in [25, 72–74]. The authors suggested to use criteria that based on both the model prediction $\hat{f}(\mathbf{x})$ and estimated standard deviation $\hat{s}(\mathbf{x})$ to identify the most promising individuals in pre-selection. This is based on the opinion that the key issue of using approximate models for evolutionary computation lies in the trade-off between the exploitation of the approximate model by sampling where it is optimized and the need to improve the approximate model by sampling where the model confidence is low [72, 74].

In 2004, Runarsson [75] proposed the so-called approximate ranking procedure to evaluate the quality of the surrogate models. The approximate ranking procedure, which is described in Algorithm 4.1, evaluates the quality of the surrogate model by its consistency in ranking the population rather than its statistical accuracy. The approximate ranking procedure in Algorithm 4.1 can be seen as an adaptive evolution control mechanism to determine the number of controlled individuals in every generation as follows: individuals

are successively selected to be evaluated according to their approximate fitness and then are added to the training set until the surrogate-based selection of the parents remains unchanged in two iteration cycles. Then, based on approximate ranking procedure and using locally weighted regression as surrogate, the local meta-model CMA-ES (lmm-CMA-ES) was firstly proposed by Kern et al. [76] and later extended for large population size by Bouzarkouna et al. [77]. The lmm-CMA-ES is a carefully designed surrogate-assisted CMA-ES algorithm.

Algorithm 4.1 Approximate Ranking Procedure

given: $\left(\mathbf{x}_k\right)_{k=1}^{\lambda}$, $f\left(\mathbf{x}\right), t, \mathcal{A}$ (archive which save all evaluated data points).
approximate: build surrogate models $\hat{f}$ based on $\mathcal{A}$, and predict $f_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$
rank and determine the parent set $\mathcal{P}_1 = \left\{\mathbf{x}_{i:\lambda}\right\}_{i=1}^{\mu}$ where $\hat{f}\left(\mathbf{x}_{1:\lambda}\right) \le \hat{f}\left(\mathbf{x}_{2:\lambda}\right) \le \cdots \le \hat{f}\left(\mathbf{x}_{\lambda:\lambda}\right)$
select the best individual: $t \leftarrow t+1$, $\mathbf{x}_t \leftarrow \underset{\mathbf{x}_{i:\lambda}}{\operatorname{argmin}}\ i \ \text{ for } \mathbf{x}_i \notin \mathcal{A}$
evaluate the selected individual $f\left(\mathbf{x}_t\right)$ and update set $\mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_t, f\left(\mathbf{x}_t\right)\right)$
for $m = 2 : \lambda$ **do**
    approximate: build surrogate $\hat{f}$ based on $\mathcal{A}$, and predict $f_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$
    determine the parent set $\mathcal{P}_m = \left\{\mathbf{x}_{i:\lambda}\right\}_{i=1}^{\mu}$ where $\hat{f}\left(\mathbf{x}_{1:\lambda}\right) \le \hat{f}\left(\mathbf{x}_{2:\lambda}\right) \le \cdots \le \hat{f}\left(\mathbf{x}_{\lambda:\lambda}\right)$
    if $\mathcal{P}_{m-1} \ne \mathcal{P}_m$ **then** (the parent set has changed)
        select an individual by $t \leftarrow t+1, \mathbf{x}_t \leftarrow \underset{\mathbf{x}_{i:\lambda}}{\operatorname{argmin}}\ i \ \text{ for } \mathbf{x}_i \notin \mathcal{A}$
        evaluate $f\left(\mathbf{x}_t\right)$ and update $\mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_t, f\left(\mathbf{x}_t\right)\right)$
    else (parent set remains unchanged)
        break (exit for loop)
    end if
    give the parent set for next generation $\mathcal{P}^{(g+1)} \leftarrow \mathcal{P}_m$
end for
output: $t, \mathcal{A}, \left(\mathbf{x}_k, f_k\right)_{k=1}^{\lambda}$

Recently, comparison-based surrogate models [78], such as ordinal regression [79] and ranking Support Vector Machine [80], are used in surrogate-assisted ES. The rank-based Support Vector Machines (SVM) was used to learn the surrogate model for CMA-ES in [78–80]. The experimental validation demonstrated the invariance of monotonous transformation of the fitness by using rank-based SVM. However, the main weakness of comparison-based surrogates assisted ES is that it can not handle well multi-modal diversity, i.e., comparison-

based surrogate models easily fail in accounting for multi-modal landscapes of fitness functions[38, 78].

4.2.2 Some Open Issues in Surrogate-Assisted ES

Even though surrogate-assisted ES has achieved considerable improvements over the past decade, there are still many open issues need to be addressed. These open issues mainly lies in i) surrogate model learning, i.e. how to use the previously evaluated data and training the surrogate model, ii) model quality measure, i.e. how to evaluate the quality of the surrogate model, and iii) model exploitation, i.e., how to exploit the built surrogate model.

Surrogate model learning is an essential step for surrogate-assisted ES. In surrogate model learning, the key task is learning one or several appropriate surrogate models from the evaluated candidates. Training data set selection has influence on the quality and the training cost of surrogate model $\hat{f}$ (besides, the modeling method also has significant on the training cost of $\hat{f}$). Thus, it is an important issue in surrogate model learning.

Another issue that relate to both design and use of surrogate models for evolutionary computation is their quality. A surrogate with poor quality may guides the search to false optimum and thus, model quality assessment is of importance.

After surrogate model has been trained, the key aspect of surrogate-assisted ES is to exploit information from the surrogate model $\hat{f}$. The two main strategies of surrogate model exploitation applied in surrogate-assisted ES are pre-selection and evolution control. There are several open questions for strategies of surrogate model exploitation, i.e., pre-selection and evolution control. The following will briefly describe some open issues of training set selection and model exploitation, which will be investigated and addressed in this work in subsequent sections.

4.2.2.1 Training Set Selection

Let n_{training} denotes the number of data points in training set $\mathcal{D}_{\text{training}}$. Three popular ways to select the training set (points/individuals for model training) are:

1. **All Previously Evaluated Points** One can use all the previously evaluated points as training set to construct surrogate models. Generally, for the same problem and using same modeling techniques, the size of training set is larger, the performance of the trained model is better. A model learned from all previously points may have better performance than one learned from just a part of existed points. However, when all the evaluated points are taken as training data, the size of training set increases during the evolutionary search, and this require higher computational cost for model training. In surrogate-assisted ES, usually, surrogate model need to retrained for large number of times, so the high training cost of the surrogate model is not desirable.

2. **Recently Evaluated Points** A simple but efficient strategy is to choose n_{training} the most recently evaluated points as training data set. This is a reasonable strategy because that the recently evaluated points are relatively close to the actual search subspace. However, in some cases, this strategy may be insufficiently "greedy" because it may "forgets" promising individuals generated more than n_{training} evaluations ago [38].

3. ***k*-Nearest Neighbor Points** Selecting evaluated points that are more relevant to current search space is reasonable. Considering that ES use Gaussian mutation, the *k*-nearest neighbor points ($k = n_{\text{training}}$) to the mean of the Gaussian distribution $\mathbf{m}$ best represent the actual search space and are good candidates for model training. The Euclidean distance is usually used as similarity measure metric. However, when f is a multi-modal function, it may happen that $\mathbf{m}$ is located around a local optimum which has been visited by the evolutionary search many generations ago, and many of n_{training} closest points to $\mathbf{m}$ are correspond to a smaller cluster of this local optimum. Thus, in this case, the built model would be very precise for the smaller cluster near the local optimum, but may not perform well in other place. This is undesirable in surrogate-assisted evolutionary search. When surrogates are built for each

individuals (such as in [81]), usually the *k*-nearest neighbor points of each individual are used to train a surrogate model for corresponding individual.

4.2.2.2 Model Quality Assessment

Without no doubt, the most popular measure for model quality is the Mean Square Error (MSE) which assesses the mean squared difference between the individual's original fitness function $f(\mathbf{x})$ and the output of the surrogate model $\hat{f}(\mathbf{x})$, that is

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i)\right)^2. \tag{4.1}$$

The MSE is evaluated over n different individuals that are taken into account for the estimation of the model quality, for instance, the $n = \lambda$ offspring individuals in one generation.

The MSE is widely used to evaluate the approximation accuracy of surrogate models. In the investigation of quality measures for surrogate model in evolutionary computation by Jin et al. [82], it has been stated that the Mean Square Error (MSE) of the model only weakly correlates with the ability to correctly select individuals in evolutionary computation, and selection based or ranking based model quality measurements were suggested to use.

From the perspective of evolutionary computation, the correct selection is of importance. In [82], Jin et al. proposed model quality measures that focus primarily on the correct model-based selection rather than the approximation accuracy of surrogate model. Some of these measures are presented below.

Let us consider the case of the (μ,λ)-selection. The model-based selection process (selection based on surrogate model) selects μ out of the λ individuals with the best predicted fitness. An individual is correctly selected, if it would also be selected by a selection process based on the original (true) fitness of the individuals. The first selection based measure is about the number of individuals that have been selected correctly using the surrogate model:

$$\rho_{\mathrm{select}} = \frac{\xi - \langle\xi\rangle}{\mu - \langle\xi\rangle}, \tag{4.2}$$

where ξ $(0 \le \xi \le \mu)$ is the number of correctly selected individuals, i.e., the number of individuals that have would also been selected if the original fitness function had been used in fitness evaluation. The expectation of ξ in case of random selection, that is

$$\langle \xi \rangle = \sum_{m=0}^{\mu} m \frac{\binom{\mu}{m}\binom{\lambda-\mu}{\mu-m}}{\binom{\lambda}{\mu}} = \frac{\mu^2}{\lambda}, \tag{4.3}$$

is used as a normalization in Equation (4.2). If all μ parents individuals are selected correctly, the measure ρ_{select} reaches its maximum value of $\rho_{\text{select}} = 1$. The negative values of ρ_{select} indicate that the selection based on the surrogate (model-based selection) is worse than a random selection. The measure ρ_{select} only evaluates the absolute number of correctly selected individuals. However, this measure does not consider the rank of the selected individuals. In case of $\rho_{\text{select}} < 1$, the measure ρ_{select} does not indicate whether the (μ+1)-th best individual or the worst individual has been selected, which may have significant influence on the evolution process. Therefore, the measure ρ_{select} is extended to include the rank of the selected individuals that are determined based on the original fitness function values.

In the extended measure $\rho_{\sim\text{select}}$, the surrogate model gets a grade of $\lambda - m$, if the *m*-th best individual based on the original fitness function is selected. Thus, the quality of the surrogate model can be indicated by summing up the grades of the selected individuals, which is demoted by $\tilde{\xi}$. Apparently, if all μ individuals are selected correctly, $\tilde{\xi}$ reaches its maximum:

$$\tilde{\xi}^{\max} = \sum_{m=1}^{\mu} (\lambda - m) = \mu\left(\lambda - \frac{\mu+1}{2}\right). \tag{4.4}$$

Similar to Equation (4.2), the measure $\rho_{\sim\text{select}}$ is defined by transforming $\tilde{\xi}$ linearly using the maximum $\tilde{\xi}^{\max}$ and the expectation $\langle \tilde{\xi} \rangle = \mu\lambda/2$ for the case of a purely random selection, expressed as:

$$\rho_{\sim\text{select}} = \frac{\tilde{\xi} - \langle \tilde{\xi} \rangle}{\tilde{\xi}^{\max} - \langle \tilde{\xi} \rangle}. \tag{4.5}$$

Moreover, Jin et al. [82] suggested to use the rank correlation coefficient [83] (also known as Spearman's rank correlation coefficient), given by

$$\rho_{\text{rank}} = 1 - \frac{6\sum_{i=1}^{\lambda} d_i^2}{\lambda\left(\lambda^2 - 1\right)}, \tag{4.6}$$

is a measure of the monotonic relation between the ranks of two variables, to evaluate the quality of the surrogate model. In the case of using ρ_{rank} to measure the monotonic relation between the individual's rank based on surrogate model and that based on original fitness function, d_i is the difference between the ranks of the *i*-th offspring (among λ) individual based on the original fitness function and on the surrogate model. The ρ_{rank} has the range $-1 \le \rho_{\text{rank}} \le 1$. The higher the value of ρ_{rank}, the stronger the monotonic relation with a positive slope between the ranks of the two variables. When the model predicts correct ranking of all λ individuals, we have $\rho_{\text{rank}} = 1$. Inversely, when the model predicts inverse ranking of individuals, $\rho_{\text{rank}} = -1$. The ρ_{rank} calculated by Equation (4.6) takes the ranks of all individuals rather than only take the ranking of the selected individuals into account.

4.2.2.3 Issues in Model Exploitation

The pre-selection and evolution control for model exploitation have already been mentioned in Section 4.2.1. As previously description, in pre-selection strategy, for each generation, $\lambda_{\text{Pre}} > \lambda$ individuals are generated (through mutation) and evaluated by surrogate model $\hat{f}$, then λ out of λ_{Pre} best individuals are selected (based on the fitness value estimated by $\hat{f}$) to re-evaluate by the original fitness function f. Thus, all the μ parent for next generation selected from λ offspring are evaluated by real fitness function f. The pre-selection strategy can be illustrated in Figure 4.1. Evolution control takes place when surrogate model $\hat{f}$ is used in fitness evaluation of ES. In individual-based evolution control, λ individuals are firstly evaluated using the surrogate $\hat{f}$, then $\lambda^* < \lambda$ individuals are selected to re-evaluated using the original fitness function f. As a result, the fitness values of μ parent for next generation can be based on surrogate $\hat{f}$ or on real fitness f. In generation-based evolution control, evolution control is carried out at a specific frequency, where the whole population of the controlled generation are evaluated using the original fitness function f, while in other generations the whole population are evaluated by surrogate $\hat{f}$. Surrogate-assisted ES using individual-based and generation-based evolution control are

illustrated in Figure 4.2. Some open issues in pre-selection and evolution control are described in the following.

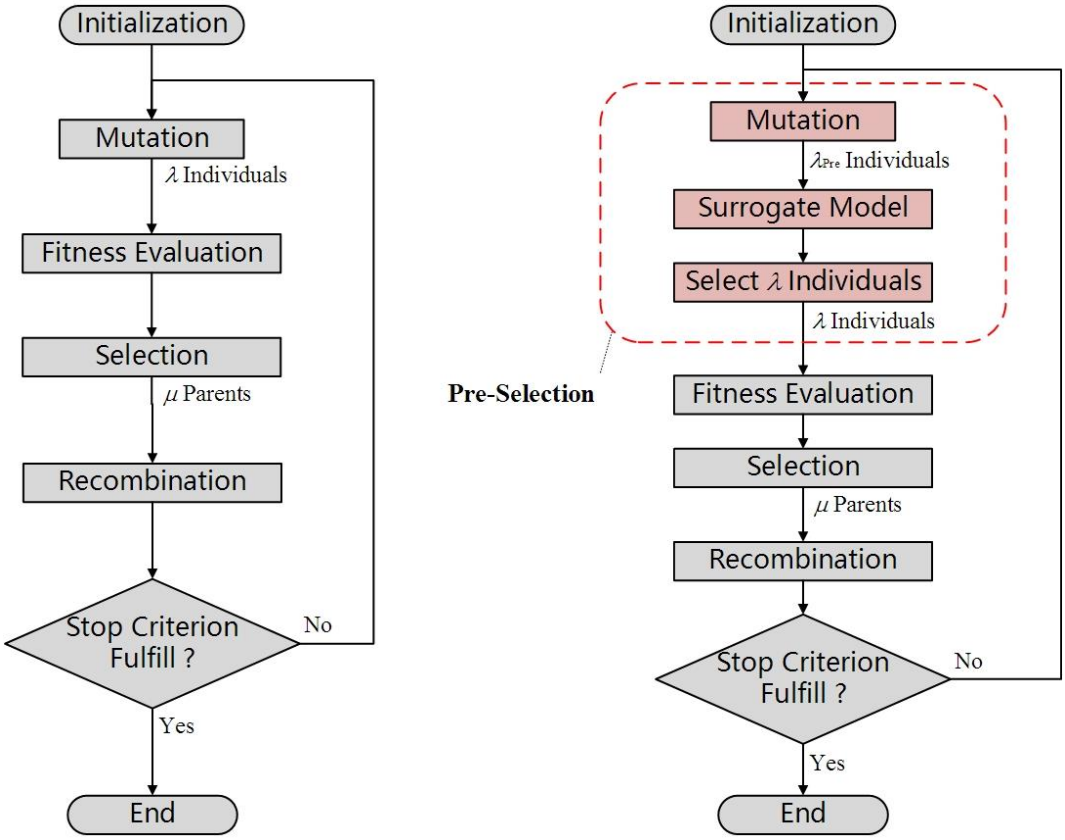

Figure 4.1 Illustration of standard (μ,λ)-ES and the surrogate-assisted ES using pre-selection.

In Pre-selection, the number λ_{Pre}, which is called the size of pre-selection population (or pre-selection population size) hereafter, has influence on evolutionary search. The size of pre-selection population λ_{Pre} controls the impact of the surrogate model $\hat{f}$ on the evolutionary optimization process. For $\lambda_{\text{Pre}} = \lambda$, the algorithm performs like a standard ES and the surrogate model $\hat{f}$ has no impact on the ES. Increasing λ_{Pre} brings about a larger selection pressure and results in a stronger impact of the model $\hat{f}$ on the convergence of the optimization process [74]. In [25, 72–74], the pre-selection population size λ_{Pre} was set as a constant during the optimization. The concept of model impact control was proposed by Ulmer et al. [84], where the value of λ_{Pre} was dynamically controlled by model quality measurement based on the number of correctly pre-selected individuals. It was demonstrated in [84] that controlling the impact of model impact (λ_{Pre}) enhances the performance of model

assisted Evolution Strategies (MAES) with fixed model influence (fixed λ_{Pre}). The concept of model impact control will attract much more attention in MAES.

Another issue in pre-selection is the pre-selection criterion which is used to identify the most promising individuals. For Gaussian process (Kriging) model assisted ES, several pre-selection criteria have been applied in literature. For other surrogate model assisted ES, usually, the model prediction is used to pre-screen the most promising individuals. However, no precise conclusion on performance of different pre-selection criteria has been drawn so far. Comprehensive study of existed pre-selection criteria is still needed. And more sophisticated pre-selection criterion may be developed in the future.

In Evolution Control, the main issues is to determine the control frequency, which denotes the number of individuals controlled in individual-based control and frequency of controlled generation in generation-based control. The control frequency plays important role in guaranteeing the correct convergence of the surrogate-assisted evolutionary optimization.

Specifically, for individual-based evolution control, in each generation η individuals are selected from λ (λ is the size of population of the standard ES) offspring for control purpose, i.e, to be evaluated by original fitness function. If the controlled individuals are chosen randomly, it is called a random strategy. If the best η individuals are selected to be controlled, we called it a best strategy. In generation-based evolution control, the whole population of η generations will be evaluated with original fitness function in every p generations, where $\eta \leq p$. Then, the control frequency in individual and generation-based evolution control can be presented by the fraction η/λ and η/p, respectively. Commonly, in evolution control, the number η is set by the users. For example, in [74] generation-based evolution control was carried out every third generation. Empirical investigation on individual-based evolution control has performed in [21], which showed that more than half of the individuals in population need to be controlled when the random selection strategy is used and about 40% of the individuals should be controlled when using best strategy, in order to guarantee a correct convergence. To manage the control frequency of generation-based control, Jin et al.

[70] introduced a framework to adapt the control frequency based on the fidelity of the surrogate model.

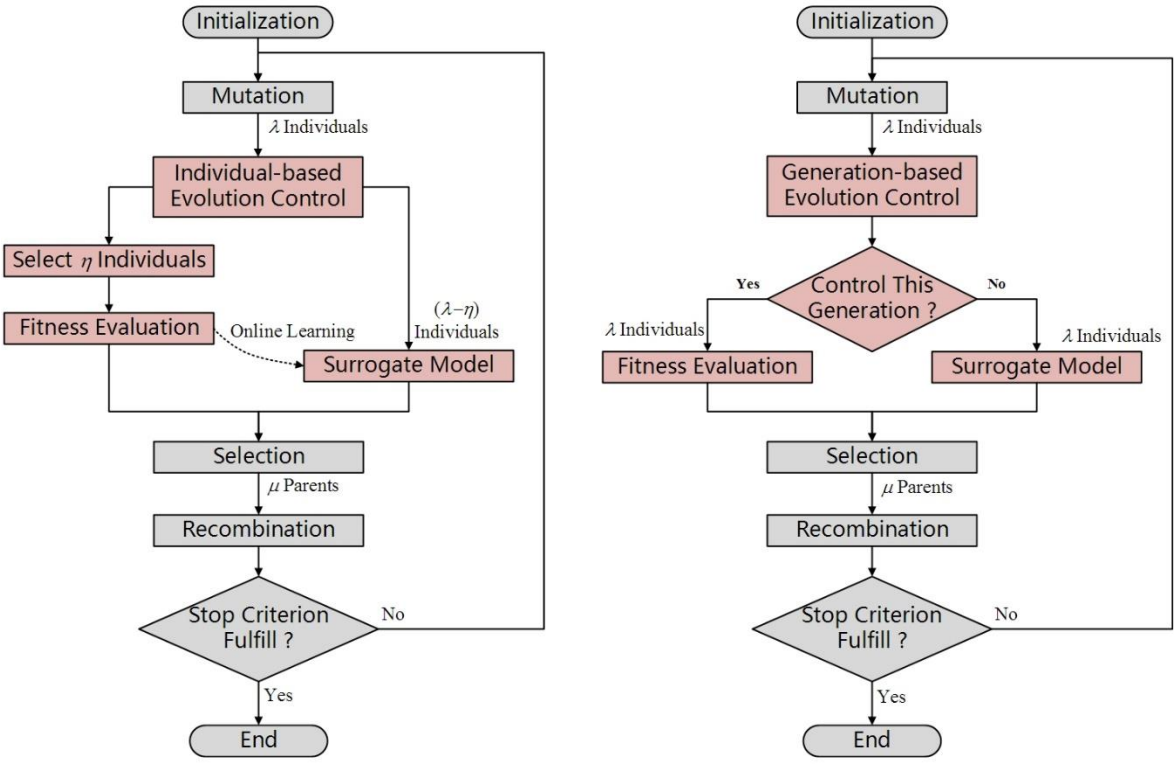

Figure 4.2 Individual-based generation-based evolution control in surrogate-assisted ES

4.3 Address Some Open Issues of Surrogate-Assisted ES

4.3.1 Training Set Selection

Training set selection affects both the quality of surrogate model learned from it and the cost of model training. Thus, it is a subject worth studying. In previous section, selecting training set as all previously evaluated points, the recently evaluated points, and nearest neighbor points are already described. In this work, we propose a new approach for training set selection based on properties of multivariate normal random variables.

Keep in mind that our goal of learning surrogate models is to predict fitness values of current offspring population. Therefore, it is rational to choose points that are relevant to current offspring population or current search subspace. The previously mentioned Recently Evaluated Points and k-Nearest Neighbor Points for training set selection are based on this

principle. However, in Recently Evaluated Points approach, no measurement is used to assess the relevance between recently evaluated points and current search subspace. So, it cannot guarantee that the selected recently evaluated points all are relevant to current search subspace. In selection of training set by the *k*-Nearest Neighbor Points, the Euclidean distance is usually used as a measurement to indicate the similarity between evaluated points and the query point in such a way that previously evaluated points with smaller Euclidean distance to the query point are chosen. For evolution strategy algorithms using isotropic mutation (uncorrelated mutation), using Euclidean distance as similarity measure metric is appropriate. When correlated mutation operator is used in evolution strategy algorithms, such as the CMA-ES, the Euclidean distance may not be useful on correlated data since there is no adjustment for the covariance. In this situation, the Mahalanobis distance becomes a more proper similarity measure metric. Therefore, we firstly suggest to use the Mahalanobis distance instead of Euclidean distance in *k*-Nearest Neighbor Points approach for training set selection and another approach of using confidence interval is proposed.

4.3.1.1 *k*-Nearest Neighbor Points to Distribution Mean based on Mahalanobis Distance

Without loss of generality, we consider the $(\mu/\mu_w,\lambda)$-CMA-ES (this algorithm has presented in Section 2.3.3) which uses a general (or correlated) mutation operator. In the evolution loop of CMA-ES, for generation g, λ offspring are generated by mutation as: $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}_d\left(\mathbf{0},\mathbf{C}^{(g)}\right)$ for $k=1,\cdots,\lambda$. Obviously, it can also be stated that the λ offspring are from a multivariate normal (MVN) distribution with mean vector $\boldsymbol{\mu}=\mathbf{m}^{(g)}$ and covariance matrix $\boldsymbol{\Sigma}=\left(\sigma^{(g)}\right)^2\mathbf{C}^{(g)}$, i.e. $\mathbf{x}_k \sim \mathcal{N}_d\left(\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$. Correspondingly, the Mahalanobis distance from a point $\mathbf{x}$ to current distribution mean $\boldsymbol{\mu}=\mathbf{m}^{(g)}$ is calculated by

$$\Delta\left(\mathbf{x},\boldsymbol{\mu}\right)=\sqrt{\left(\mathbf{x}-\boldsymbol{\mu}\right)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)}=\sqrt{\left(\mathbf{x}-\mathbf{m}^{(g)}\right)^{\mathrm{T}}\left(\left(\sigma^{(g)}\right)^2\mathbf{C}^{(g)}\right)^{-1}\left(\mathbf{x}-\mathbf{m}^{(g)}\right)}. \qquad (4.7)$$

Generally speaking, data points with smaller Mahalanobis distance to current mean of mutation distribution are more relevant to current offspring population. Thus, we can select

the *k*-Nearest Neighbor Points according to the Mahalanobis distance to current distribution mean $\mathbf{m}^{(g)}$ as the training set.

4.3.1.2 Confidence Interval for Training Set Selection

Considering the λ offspring are from a MVN distribution with mean vector $\boldsymbol{\mu} = \mathbf{m}^{(g)}$ and covariance matrix $\boldsymbol{\Sigma} = \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}$, i.e. $\mathbf{x}_k \sim \mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$, the actual (current) search subspace (the region that the distributed offspring can reach) is determined by current distribution of offspring $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$. It is apparent that the sampled offspring probably lie in the region with high probability of the multivariate normal distribution. Consequently, the region yielded by the confidence interval of the multivariate normal distribution can be used to represent the actual subspace with a given probability.

The confidence interval for the multivariate normal distribution $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ yields a region consisting of those vectors $\mathbf{x} \in \mathbb{R}^d$ satisfying

$$\left(\mathbf{x} - \boldsymbol{\mu}\right)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \left(\mathbf{x} - \boldsymbol{\mu}\right) \le \chi_d^2\left(p\right), \tag{4.8}$$

where $\chi_d^2\left(p\right)$ is the quantile function for probability p $\left(0 < p < 1\right)$ of the chi-squared distribution with d degrees of freedom. The inequality (4.8) provides the confidence region containing p of the probability mass of the MVN distribution $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$. In other words, a random vector from distribution $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ has probability p of satisfying the inequality (4.8).

Since λ offspring $\mathbf{x}_k$, $k = 1, \cdots, \lambda$ are from MVN distribution $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ (where $\boldsymbol{\mu} = \mathbf{m}^{(g)}$ and $\boldsymbol{\Sigma} = \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}$), the region defined by inequality (4.8) has a probability p that it cover the offspring. Put another way, this region is the subspace that the offspring generated from the MVN distribution $\mathcal{N}_d\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ can reach with a probability p. The probability p indicates the extent that the confidence region covers the actual search subspace. Therefore, we propose to select the training data set $\mathcal{D}_{\text{training}}$ by

$$\mathcal{D}_{\text{training}} = \left\{ \left(\mathbf{x}_i, f\left(\mathbf{x}_i\right)\right) \in \mathcal{A} \,\middle|\, \left(\mathbf{x}_i - \mathbf{m}^{(g)}\right)^{\mathrm{T}} \left(\left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right)^{-1} \left(\mathbf{x}_i - \mathbf{m}^{(g)}\right) \le \chi_d^2\left(p_{\text{training}}\right) \right\}, \tag{4.9}$$

where $\mathcal{A}$ is a set stores all the previously evaluated points (individuals, candidates) and associated fitness function values, i.e. $\mathcal{A} = \left\{\left(\mathbf{x}_i, f\left(\mathbf{x}_i\right)\right), i = 1, \cdots, n_{\text{evaluated}}\right\}$ ($n_{\text{evaluated}}$ is the number of evaluated points), $\chi_d^2\left(p_{\text{training}}\right)$ is the quantile function for probability p_{training} of the chi-squared distribution with d degrees of freedom, and the probability p_{training} signifies the confidence of selected training set $\left(0 < p_{\text{training}} < 1\right)$. When we use a large p_{training}, evaluated points which lie in an interval with high confidence are selected to form the training set $\mathcal{D}_{\text{training}}$. While small p_{training} is used, the training set $\mathcal{D}_{\text{training}}$ only select evaluated points contained in a low confidence interval. Usually, $\mathcal{D}_{\text{training}}$ selected with large p_{training} could include more points than that with small p_{training}. The value of p_{training} is set by the user and can be chosen as 95.45% or 99.73% or any other values. The surrogate models build on $\mathcal{D}_{\text{training}}$ selected by Equation (4.9) are local models around the distribution mean $\mathbf{m}^{(t)}$ for current offspring population.

4.3.2 Pre-Selection Strategy with Model Impact Control

4.3.2.1 Introduction of Pre-Selection with Model Impact Control (CPS)

The size of pre-selection population size λ_{Pre} controls the impact of the surrogate model $\hat{f}$ on the evolutionary optimization process. Thus, the search procedure would benefit from the appropriate control of λ_{Pre}. Simply, the pre-selection population size is set as a constant by users during the optimization. Recently, researchers attempt to adapt λ_{Pre} during the evolutionary search process. It is intuitive to use the model quality to control the pre-selection population size λ_{Pre}. In [84], the value of λ_{Pre} was controlled by model quality measurement based on the number of correctly pre-selected individuals. This idea gives significant guidance for us in pre-selection population size control.

In this paragraph, we briefly present the model quality measurement and adaptation of λ_{Pre} that were proposed by Ulmer et al. [84]. Since in optimization by evolution strategies, only the correct selection is of importance, the authors used a selection based model quality which was similar to that proposed by Jin et al. [82]. Consider the model based selection process selects μ out of λ individuals with the best predicted fitness. An individual is

correctly selected, if it would also be selected by a selection process based on the original (true) fitness of the individuals. Thus, the number of correctly selected individuals can be used to measure the quality of the model based selection process. The authors assumed that the selection quality determined for a selection of μ out of λ individuals, for which we known the predicted and the true fitness, is equivalent to the selection quality of the pre-selection. If the individual with the *i*-th best true fitness is correctly selected, a rank (grade) of $(\lambda - i)$ is given to this individual (this rank could be considered as the grade of the individual in selection). The summed rank of all correctly selected individuals is defined as the quality $Q_{\text{selection}}$ of the model based selection process. The difference between $Q_{\text{selection}}$ and $\tilde{\xi}$ in Equation(4.4) is that $Q_{\text{selection}}$ only counts the grades of correctly selected individuals but $\tilde{\xi}$ counts the grades of all selected individuals including both correctly and incorrectly selected individuals. When none individual is correctly selected, $Q_{\text{selection}}$ has its minimum value $\left(Q_{\text{selection}}^{\min} = 0\right)$. If all individuals are correctly selected, the maximum model quality is

$$Q_{\text{selection}}^{\max} = \sum_{i=1}^{\mu}(\lambda - i) = \mu\lambda - \frac{\mu(\mu+1)}{2}. \tag{4.10}$$

The expectation value of $Q_{\text{selection}}$ for a purely random selection process is given as the product of the expectation value of the number of correctly selected individuals and the expectation value of the rank of a correctly selected individual :

$$Q_{\text{selection}}^{\text{rand}} = \left(\sum_{i=1}^{\mu} i \frac{\binom{\mu}{i}\binom{\lambda-\mu}{\mu-i}}{\binom{\lambda}{\mu}} \right) \cdot \left(\frac{1}{\mu} \sum_{i=1}^{\mu}(\lambda - i) \right) = \frac{\mu^2}{\lambda} \cdot \frac{2\lambda - \mu - 1}{2}. \tag{4.11}$$

After each generation t the actual model quality $Q_{\text{selection}}^{(g)}$ is evaluated and compared to the expected quality of the random selection process $Q_{\text{selection}}^{\text{rand}}$. For $Q_{\text{selection}}^{(g)} > Q_{\text{selection}}^{\text{rand}}$ the model based selection is better than a random selection, and λ_{Pre} should be increased. On the contrary, for $Q_{\text{selection}}^{(g)} < Q_{\text{selection}}^{\text{rand}}$ the λ_{Pre} should be decreased. When $Q^{(g)} = Q^{\text{rand}}$ the value of λ_{Pre} is kept. Therefore, the update procedure for λ_{Pre} is the following:

$$\lambda_{\text{Pre}}^{(g+1)} = \begin{cases} \lambda_{\text{Pre}}^{(g)} + \dfrac{Q_{\text{selection}}^{(g)} - Q_{\text{selection}}^{\text{rand}}}{Q_{\text{selection}}^{\text{max}} - Q_{\text{selection}}^{\text{rand}}} \delta_{\text{Pre}}, & \text{if } Q_{\text{selection}}^{(g)} > Q_{\text{selection}}^{\text{rand}} \\ \lambda_{\text{Pre}}^{(g)} - \dfrac{Q_{\text{selection}}^{\text{rand}} - Q_{\text{selection}}^{(g)}}{Q_{\text{selection}}^{\text{rand}}} \delta_{\text{Pre}}, & \text{otherwise} \end{cases} \tag{4.12}$$

where δ_{Pre} is the adaptation rate. It was demonstrated in [84] that controlling the impact of model impact (λ_{Pre}) enhances the performance of model assisted Evolution Strategies (MAES) with fixed model influence (fixed λ_{Pre}).

4.3.2.2 Proposed Pre-Selection with Model Impact Control

Proposed Selection-based Model Quality Measure using Recombination Weights

In evolutionary algorithms, the selection operator choosing the best individuals to enter next evolution loop and thus gives the evolutionary search a direction. Specially, in evolution strategies, the selection is based on fitness ranks and deterministic. The recombination after selection is also deterministic in ES. Thus, selection is particularly important in evolution strategies. Consequently, in surrogate-assisted evolutionary computation, the quality of model-based selection (selection by using surrogate model) is of importance, particularly in surrogate-assisted evolution strategies.

In the investigation of quality measures for surrogate model in evolutionary computation by Jin et al. [82], it has been stated that the the Mean Square Error (MSE) of the model only weakly correlates with the ability to correctly select individuals in evolutionary computation, and selection based or ranking based model quality measurements were suggested to use. In [84], a selection based model quality was used to control the size of pre-selection population for surrogate-assisted evolution strategies. This method for updating λ_{Pre} has pointed out a direction for pre-selection population size control, i.e., dynamically controlling λ_{Pre} based on model quality. In this thesis, a variant of the selection based model quality measurement, which was proposed by Ulmer et al. in [84] and described above, is defined based on the weights used in recombination operator in an evolution strategy.

We consider an evolution strategy that use weighted recombination operator, denoted by $(\mu/\mu_w,\lambda)$-ES. In weighted recombination, the weight values depend on the fitness ranking, in that better parents get larger weights than inferior ones. The widely used super-linear decrease weights are given by

$$\begin{aligned} & w_k = \frac{w_k'}{\sum_{i=1}^{\mu} w_i'}, \qquad \text{for } k = 1,\cdots,\mu \\ & \text{with } w_k' = \ln\left(\frac{\lambda+1}{2}\right) - \ln\left(\text{rank}\left(f\left(\mathbf{x}_k\right)\right)\right), \qquad \text{for } k = 1,\cdots,\mu \end{aligned} \tag{4.13}$$

where $\text{rank}\left(f\left(\mathbf{x}_k\right)\right)$ is the fitness ranking of individual $\mathbf{x}_k$ in all λ offspring $\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_\lambda$. The weighted recombination generates a single solution vector $\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$, which is the new mean vector of mutation distribution. It apparent that individuals with high fitness ranking have larger weights and thus contribute more to the search step.

Considering the model based selection process selecting μ out of the λ individuals with the best predicted fitness, an individual is correctly selected, if it would also be selected by a selection process based on the original (true) fitness of the individuals. Usually, the number of correctly selected individuals can be used to measure the quality of the surrogate model. We define a selection based model quality that takes in account not only the number of correctly selected individuals but also the real fitness ranking of individuals. Our idea is that, for each model-based selected individual (individual that is selected according to their prediction fitness), if it is also selected by selection process using real fitness, its associated recombination weights calculated by real fitness ranking is assigned to this individual as the score it obtains in model selection; otherwise, it has score zero. The measurement of selection based model quality Q_w is defined as the summed score of all model selected individuals. Obviously, a larger value of Q_w means the surrogate model has higher quality of correctly selecting individuals, and vice versa. If the model selects none correct individual, Q_w reaches its minimum value $Q_w^{\min} = 0$. When all individuals are correctly selected, Q_w takes its maximum value $Q_w^{\max} = \sum_{i=1}^{\mu} w_i = 1$. In other cases, i.e., only part of model selected individuals are correct, $0 < Q_w < 1$. From the definition of Q_w, it can be found that when

individuals with higher real fitness ranking and/or more individuals are selected correctly, Q_w obtains higher value and the model quality is better.

Similar to Equation (4.11), the expectation value of Q_w for a purely random selection process is simply given as the product of the expectation value of the number of correctly selected individuals and the expectation value of the score of a correctly selected individual :

$$Q_w^{\text{rand}} = \left(\sum_{i=1}^{\mu} i \frac{\binom{\mu}{i}\binom{\lambda-\mu}{\mu-i}}{\binom{\lambda}{\mu}} \right) \cdot \left(\frac{1}{\mu} \sum_{i=1}^{\mu} w_i \right) = \frac{\mu^2}{\lambda} \cdot \frac{1}{\mu}. \tag{4.14}$$

In surrogate-assisted evolution strategies using pre-selection strategy, to measure the quality of the pre-selection process with above defined Q_w, we need to know the true fitness of all λ_{Pre} individuals. However, only the true fitness of λ most promising pre-selected individuals is known. We also taken the assumption in [84] that the model quality Q_w for a selection of μ out of the λ individuals, for which we know the predicted and the true fitness, is equivalent to that of the pre-selection process.

Controlling the Size of Pre-selection Population (Model Impact) λ_{Pre}

The evolutionary search can benefit from the surrogate model assistance if the model selection process performs better than a purely random selection and correspondingly model selection quality is better than the quality of a purely random selection.

After each generation in the surrogate-assisted ES with pre-selection, the actual measured selection quality $Q_w^{(g)}$ is computed and compared with the expected quality of the random selection $Q_{\text{w}}^{\text{rand}}$ (given in (4.14)).

If $Q_w^{(g)} > Q_w^{\text{rand}}$ the model based selection is better than a random selection, and λ_{Pre} should be increased. On the contrary, for $Q_{\text{w}}^{(g)} < Q_{\text{w}}^{\text{rand}}$ the λ_{Pre} should be decreased. When $Q_{\text{w}}^{(g)} = Q_{\text{w}}^{\text{rand}}$ the value of λ_{Pre} is kept. Therefore, λ_{Pre} is adapted similarly as (4.12), i.e.,

$$\lambda_{\text{Pre}}^{(g+1)} = \begin{cases} \lambda_{\text{Pre}}^{(g)} + \dfrac{Q_w^{(g)} - Q_w^{\text{rand}}}{Q_w^{\max} - Q_w^{\text{rand}}} \delta_{\text{Pre}}, & \text{if } Q_w^{(g)} > Q_w^{\text{rand}} \\ \lambda_{\text{Pre}}^{(g)} - \dfrac{Q_w^{\text{rand}} - Q_w^{(g)}}{Q_w^{\text{rand}}} \delta_{\text{Pre}}, & \text{otherwise} \end{cases} \tag{4.15}$$

where $\delta_{\lambda_{\text{Pre}}}$ is the adaptation rate, which is set by the user.

Model Impact Control based on Different Model Quality Measures

In pre-selection with model impact control, different model quality measures can be used. In our work, three model quality measures are included. These three model quality measures are: 1) proposed selection-based model quality measure using recombination weights Q_w; 2) selection-based model quality measure $Q_{\text{selection}}$ proposed by Ulmer et al. [84]; 3) rank correlation coefficient ρ_{rank} proposed by Jin et al. [82]. The strategy of updating the pre-selection population size λ_{Pre} is expressed by Equation (4.12) or (4.15). In the model impact control rule, three critical values of model quality measures and the model quality evaluated at current generation are used. These three critical values are the minimum, maximum and a threshold value of model quality, which are respectively noted by $Q^{\min}$, $Q^{\max}$ and Q^{TS}. The size of pre-selection population λ_{Pre} is increased when the model quality of current generation is larger than the threshold value, i.e., $Q^{(g)} > Q^{\text{TS}}$; and λ_{Pre} is decreased if $Q^{(g)} < Q^{\text{TS}}$. The uniform formulation of model impact control based on model quality measure Q, which includes Q_w, $Q_{\text{selection}}$ and ρ_{rank}, can be given as:

$$\lambda_{\text{Pre}}^{(g+1)} = \begin{cases} \lambda_{\text{Pre}}^{(g)} + \dfrac{Q^{(g)} - Q^{\text{TS}}}{Q^{\max} - Q^{\text{TS}}} \delta_{\text{Pre}}, & \text{if } Q^{(g)} > Q^{\text{TS}} \\ \lambda_{\text{Pre}}^{(g)} - \dfrac{Q^{\text{TS}} - Q^{(g)}}{Q^{\text{TS}} - Q^{\min}} \delta_{\text{Pre}}, & \text{otherwise} \end{cases} \tag{4.16}$$

The critical values of model quality measures are listed in Table 4.1. For model quality measure ρ_{rank}, $\rho_{\text{rank}} = 0.5$ indicates a moderate monotonic relationship between the predicted fitness and exact fitness. Thus, we choose $\rho_{\text{rank}} = 0.5$ as the threshold value of ρ_{rank}.

Table 4.1 Critical values of model quality measures

Model quality measure Q	Minimum value $Q^{\min}$	Maximum value $Q^{\max}$	Threshold value Q^{TS}
Q_w	0	1	μ/λ
$Q_{\text{selection}}$	0	$\mu\lambda-\mu(\mu+1)/2$	$(\mu^2/\lambda)\cdot(2\lambda-\mu-1)/2$
ρ_{rank}	-1	1	0.5

4.3.2.3 Algorithm of Pre-Selection Strategies

The pre-selection concept has been introduced in Section 4.2.1. In pre-selection procedure, $\lambda_{\text{Pre}} > \lambda$ individuals are generated, then all λ_{Pre} individuals are evaluated by surrogate model $\hat{f}$ and the estimated fitness values are used to pre-select the λ best individuals, which will be evaluated by the original fitness function. After the initial sampling, the pre-selection procedure can be performed in each iteration of the evolution loop. The module of pre-selection for evolution loop is expressed in Algorithm 4.2.

Algorithm 4.2 Pre-Selection Procedure

1: **given:** $\mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \mathcal{A}, \lambda_{\text{Pre}}^{(g)}$
2: training set selection: $\mathcal{D}_{\text{T}} \leftarrow \text{training\_set\_selection}(\mathcal{A})$
3: model training: $\hat{f} \leftarrow \text{model\_training}(\mathcal{D}_{\text{T}})$
4: **for** $k = 1, \cdots, \lambda_{\text{Pre}}^{(g)}$ **do**
5: $\quad \mathbf{s}_k = \mathcal{N}_d(\mathbf{0}, \mathbf{I})$
6: $\quad \mathbf{y}_k = \mathbf{m}^{(g)} + \sigma^{(g)} \left(\mathbf{C}^{(g)}\right)^{1/2} \mathbf{s}_k$ // $\mathbf{y}_k$ represents individuals in pre-selection population
7: $\quad \hat{f}_k = \hat{f}(\mathbf{y}_k)$
8: **end for**
9: select promising individuals according to $\hat{f}_k$: $\mathbf{x}_k = \mathbf{y}_{k:\lambda_{\text{pre}}}$, $\mathbf{z}_k = \mathbf{s}_{k:\lambda_{\text{pre}}}$ for $k = 1, \cdots, \lambda$.
10: **output:** $\left(\mathbf{x}_k, \mathbf{z}_k, \hat{f}_k\right)_{k=1}^{\lambda}$ //offspring for re-evaluation using original fitness function

In Algorithm 4.2, the current distribution mean vector $\mathbf{m}^{(g)}$, step-size $\sigma^{(g)}$, covariance matrix $\mathbf{C}^{(g)}$,and the archive $\mathcal{A}$ which contains all previously evaluated data points are used in pre-selection (Line 1). The pre-selection population size for current generation $\lambda_{\text{Pre}}^{(g)}$ is the parameter of pre-selection procedure. With the inputs, training set $\mathcal{D}_{\text{T}}$ is firstly selected by the training set selection method (Line 2), which can be the recently evaluated points, *k*-nearest neighbor points to distribution mean and confidence interval method proposed in Section 4.3.1, and the surrogate model $\hat{f}$ is trained from the selected training set (Line 3).

Then, $\lambda_{\text{Pre}}^{(g)}$ individuals are generated and subsequently evaluated by the surrogate (Line 4~8). According to the estimated fitness values, λ best individuals are selected for re-evaluation using the original fitness function (Line 9). The λ pre-selected offspring are exported for original fitness evaluation and strategy parameters adaptation.

The size of pre-selection population λ_{Pre} can be constant or be controlled during the evolution loop. For pre-selection without model impact control (PS), the pre-selection population λ_{Pre} keeps constant during the evolution loop, i.e., $\lambda_{\text{Pre}}^{(1)} = \lambda_{\text{Pre}}^{(2)} = \cdots = \lambda_{\text{Pre}}^{(g)}$. While, for pre-selection with model impact control (CPS), λ_{Pre} is updated during evolutionary search according to the model quality. The updating rule of λ_{Pre} is given in Equation (4.16) based on the model quality measures including Q_w, $Q_{\text{selection}}$ and ρ_{rank} have been described previously.

4.3.3 Individual-based Evolution Control

In surrogate-assisted evolution strategies with evolution control, the control frequency, which is described in Section 4.2.2.3, is of significance. On one hand, in order to prevent the evolutionary search from being misled by a false optimum introduced by the surrogate model [85], the surrogate model should be used together with the original fitness function. On other hand, the original fitness function should be used sparsely such that computational cost on (original) fitness evaluation is saved as much as possible. Evolution control could be regarded as a scheme to make balance between saving the computational cost of fitness evaluation and ensuring the evolutionary search converging to the global optimum or a near-optimum of the original fitness function. This subsection and next subsection devote to putting forward strategies for individual-based and generation-based evolution control, respectively.

The concepts of evolution control have already introduced in Section 1.3.3. In surrogate-assisted evolutionary computation, the evolution control means that the original fitness function is used to evaluate some/all individuals in some/all generations. An individual that is evaluated using original fitness function is called a controlled individual. Similarly, a generation in which all its individuals are evaluated using the original fitness

function is called a controlled generation. Correspondingly, evolution control can be divided into two approaches: individual-based and generation-based evolution control. In individual-based evolution control, a certain number of individuals within a generation are selected to be evaluated using the original fitness function, and other individuals in the generation use the surrogate model for fitness evaluation. Generation-based evolution control is carried out at a certain frequency. In a controlled generation, all individuals are evaluated with the original fitness function. The most important question for evolution control, both of individual-based and generation-based evolution control, is how to determine the control frequency, which signifies the number of controlled individuals and generations for individual-based and generation-based evolution control, respectively, in order to guarantee the correct convergence of the evolutionary search when false optima are present in the surrogate model of fitness function.

For individual-based evolution control, both fixed and adaptive control methods can be used. In fixed individual-based evolution control, the number of controlled individuals is fixed, i.e., a fixed number of individuals are selected to re-evaluated by using the original fitness function. While, for adaptive individual-based evolution control, the number of re-evaluated individuals is adaptive, usually depend on the fidelity (quality) of the surrogate model. The controlled individuals can be selected randomly or by a best strategy, which selects best individuals based on predicted fitness. Obviously, fixed individual-based control is simpler and thus more easily to implement than adaptive control. However, adaptive individual-based evolution control may require less number of re-evaluated individuals owing to the adaptive control frequency, and thus may save more computational cost on fitness evaluation and consequently would be more efficient.

4.3.3.1 Fixed Individual-based Control (FIC) using Metric

In individual-based control (FIC), usually, the individuals with best predicted fitness are selected to be controlled. Compared with other surrogate models like RBF, MLP model, the advantage of Kriging model (or Gaussian process model) is that it not only predicts fitness value but also provides the standard deviation for the predicted fitness without

additional computational cost. In this work, we comprehensively investigate the criteria which based on the model prediction $\hat{f}(\mathbf{x})$ or/and estimated standard deviation $\hat{s}(\mathbf{x})$ to select individuals for re-evaluation. The criteria used to select individuals which will be controlled are called metrics hereafter in this thesis. Five different metrics, which have been used in surrogate-based optimization [54, 57] but relatively new in surrogate-assisted evolution computation, including the mean of model prediction (Mean), estimated standard deviation (SD) of prediction, statistical lower bound (SLB), probability of improvement (POI), and expected improvement (EI), are introduced and studied in KA-CMA-ES using fixed individual-based control. These five metrics are defined by

$$C_{\text{Mean}}(\mathbf{x}) = \hat{f}(\mathbf{x}) \tag{4.17}$$

$$C_{\text{SD}}(\mathbf{x}) = \hat{s}(\mathbf{x}) \tag{4.18}$$

$$C_{\text{SLB}}(\mathbf{x}) = \hat{f}(\mathbf{x}) - A \cdot \hat{s}(\mathbf{x}) \tag{4.19}$$

$$C_{\text{POI}}(\mathbf{x}) = \begin{cases} \Phi\left(\dfrac{f_{\min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) & \text{if } \hat{s}(\mathbf{x}) > 0 \\ 0 & \text{if } \hat{s}(\mathbf{x}) = 0 \end{cases} \tag{4.20}$$

$$C_{\text{EI}}(\mathbf{x}) = \begin{cases} \left(f_{\min} - \hat{f}(\mathbf{x})\right)\Phi\left(\dfrac{f_{\min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\phi\left(\dfrac{f_{\min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), & \text{if } \hat{s}(\mathbf{x}) > 0 \\ 0, & \text{if } \hat{s}(\mathbf{x}) = 0 \end{cases} \tag{4.21}$$

where A is a constant and in this work we set $A = 2$, $\phi(\cdot)$ and $\Phi(\cdot)$ are respectively the probability distribution function and cumulative distribution function of standard normal distribution, and $f_{\min}$ is the current minimal fitness function value.

In fixed individual-based control, a fixed fraction of the individuals ($\eta \cdot \lambda$) in each generation are selected to be controlled according to above five metrics. Specifically, for fixed individual-based control using metric Mean, $\eta \cdot \lambda$ individuals which have the smaller predicted mean fitness are selected for controlling. When the SD metric is used, individuals with larger estimated standard deviation are controlled. If the SLB is adopted, individuals with the smaller SLB values are controlled. For POI and EI metrics, individuals that have

the larger POI or EI values are selected and then controlled. Above described fixed individual-based control (FIC) can be illustrated by Algorithm 4.3.

Empirical investigations on evolutionary computation with approximate fitness functions [21] showed that the number of the controlled individuals should be larger than 50% of the population size if individual-based control is used and that the best strategy was recommended by the authors. The best strategy based on metric $C(\mathbf{x})$ (including Mean, SD, SLB, POI and EI), which have been given from Equation (4.17) to (4.21), is used in fixed individual-based control to select the controlled individuals. As for the fraction of controlled individuals, we set $\eta = 0.5$, i.e., half of the individuals in each generation is controlled.

Algorithm 4.3 Fixed Individual-based Control (FIC) using Metric

given: $(\mathbf{z}_k, \mathbf{x}_k)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, t, \mathcal{A}, f(\mathbf{x}), \eta, C(\mathbf{x})$
training set selection: $\mathcal{D}_{\mathrm{T}} \leftarrow$ training_set_selection$(\mathcal{A})$
model training: $\hat{f} \leftarrow$ model_training$(\mathcal{D}_{\mathrm{T}})$
for $k = 1, \cdots, \lambda$ **do**
    $C_k \leftarrow$ metric $C(\mathbf{x}_k)$ computing by model $\hat{f}$ //compute the metric values of individuals
end for
$\mathcal{P}_{\text{controlled}} = \{\mathbf{x}_{1:\lambda}, \mathbf{x}_{2:\lambda}, \cdots, \mathbf{x}_{\lfloor \eta\lambda \rfloor:\lambda}\}$, $\mathcal{P}_{\text{uncontrolled}} = \{\mathbf{x}_{\lfloor \eta\lambda \rfloor+1:\lambda}, \cdots, \mathbf{x}_{\lambda:\lambda}\}$ //best strategy based on $C(\mathbf{x})$
for $\mathbf{x}_k \in \mathcal{P}_{\text{controlled}}$ **do**
    $f_k = f(\mathbf{x}_k)$ //evaluated by original fitness function
    $t \leftarrow t + 1$
    $\mathcal{A} \leftarrow \mathcal{A} \cup (\mathbf{x}_k, f_k)$
    $\mathcal{D}_{\mathrm{T}} \leftarrow \mathcal{D}_{\mathrm{T}} \cup (\mathbf{x}_k, f_k)$
end for
$\hat{f} \leftarrow$ train_model$(\mathcal{D}_{\mathrm{T}})$ //update surrogate model
for $\mathbf{x}_k \in \mathcal{P}_{\text{uncontrolled}}$ **do**
    $f_k = \hat{f}(\mathbf{x}_k)$ //evaluated by surrogate model
end for
output: $t, \mathcal{A}, (\mathbf{x}_k, \mathbf{z}_k, f_k)_{k=1}^{\lambda}$

4.3.3.2 Mixed Individual-based Control (MIC)

With the efforts of drawing the advantages of both fixed and adaptive individual-based evolution control, an individual-based control strategy called mixed Individual-based control

(MIC) which combines both features of fixed and adaptive control in some way, is proposed and presented below. Similarly to fixed individual-based control, in this mixed individual-based evolution control strategy, $\eta \cdot \lambda$ individuals are chosen by a best strategy for re-evaluation in each generation. However, the value of η is vary between the user defined lower bound $\eta_{\min}$ and upper bound $\eta_{\max}$, i.e. $0 < \eta_{\min} \le \eta \le \eta_{\max} < 1$, which is mainly different from the FIC. In each generation, $\eta_{\min} \cdot \lambda$ individuals selected by a best strategy are unconditionally re-evaluated using original fitness function $f(\mathbf{x})$. Up to $(\eta_{\max} - \eta_{\min})\lambda$ more offspring which take on better fitness values (estimated by surrogates $\hat{f}(\mathbf{x})$) than the current best solution can be re-evaluated too. With these newly evaluated points, the surrogate model is retrained (updated) and the fitness values of uncontrolled individuals are re-estimated by the updated surrogate model.

Furthermore, since Kriging model is used as the surrogate model, a metric function $C(\mathbf{x})$ that is based on both the model prediction $\hat{f}(\mathbf{x})$ and estimated standard deviation $\hat{s}(\mathbf{x})$ to identify the most promising individuals whose fitness are probably better than the current best fitness. Evidently, besides the mean of prediction (Mean), the statistical lower bound (SLB) can be used in above described mixed individual-based control without any other change of the control method.

Above described mixed individual-based control strategy can be regarded as a combination of fixed control with control frequency of $\eta_{\min} \cdot \lambda$ and an adaptive control that controls up to $(\eta_{\max} - \eta_{\min})\lambda$ offspring individuals that has predicted fitness better than current best solution. In addition, after the re-evaluation of controlled individuals, on-line model updating is performed by retraining the surrogate model based on training set that adds these newly evaluated points. This could improve the quality of surrogate model and is good to the evolutionary computation.

Above proposed mixed individual-based control (MIC) is implemented here and given in Algorithm 4.4. The metric Mean and SLB can be used in MIC. The lower bound $\eta_{\min}$ and upper bound $\eta_{\max}$ are set as $\eta_{\min} = 0.4$ and $\eta_{\max} = 0.8$. This means that at least 40% and up to 80% individuals are controlled in each generation. Too small $\eta_{\min}$ may lead to the algorithm

fail of converging to the global optimum and too large $\eta_{\min}$ would require more exact fitness function evaluations.

Algorithm 4.4 Mixed Individual-based Control (MIC)

given: $(\mathbf{z}_k, \mathbf{x}_k)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, t, \mathcal{A}, f(\mathbf{x}), \eta_{\min}, \eta_{\max}, C(\mathbf{x})$
training set selection: $\mathcal{D}_{\mathrm{T}} \leftarrow \text{training\_set\_selection}(\mathcal{A})$
model training: $\hat{f} \leftarrow \text{model\_training}(\mathcal{D}_{\mathrm{T}})$
for $k = 1, \cdots, \lambda$ **do**
    $C_k \leftarrow$ metric $C(\mathbf{x}_k)$ computing by model $\hat{f}$ //compute the metric values of individuals
end for
$\mathcal{P}_{\text{controlled}} = \{\mathbf{x}_{1:\lambda}, \mathbf{x}_{2:\lambda}, \cdots, \mathbf{x}_{\lfloor \eta_{\min}\lambda \rfloor:\lambda}\}$, $\mathcal{P}_{\text{uncontrolled}} = \{\mathbf{x}_{\lfloor \eta_{\min}\lambda \rfloor+1:\lambda}, \cdots, \mathbf{x}_{\lambda:\lambda}\}$ //best strategy using $C(\mathbf{x})$
for $k = (\lfloor \eta_{\min}\lambda \rfloor + 1), \cdots, \lfloor \eta_{\max}\lambda \rfloor$ **do**
    if $C_{k:\lambda} < f_{\min}$ **then** // $f_{\min}$ is the minimum fitness in $\mathcal{A}$
        $\mathcal{P}_{\text{controlled}} \leftarrow \mathcal{P}_{\text{controlled}} \bigcup \mathbf{x}_{k:\lambda}$, $\mathcal{P}_{\text{uncontrolled}} = \{\mathbf{x}_{k+1:\lambda}, \cdots, \mathbf{x}_{\lambda:\lambda}\}$
    end if
end for
for $\mathbf{x}_k \in \mathcal{P}_{\text{controlled}}$ **do**
    $f_k = f(\mathbf{x}_k)$ //evaluated by original fitness function
    $t \leftarrow t + 1$
    $\mathcal{A} \leftarrow \mathcal{A} \bigcup (\mathbf{x}_k, f_k)$
    $\mathcal{D}_{\mathrm{T}} \leftarrow \mathcal{D}_{\mathrm{T}} \bigcup (\mathbf{x}_k, f_k)$
end for
$\hat{f} \leftarrow \text{train\_model}(\mathcal{D}_{\mathrm{T}})$ //update surrogate model
for $\mathbf{x}_k \in \mathcal{P}_{\text{uncontrolled}}$ **do**
    $f_k = \hat{f}(\mathbf{x}_k)$ //evaluated by surrogate model
end for
output: $t, \mathcal{A}, (\mathbf{x}_k, \mathbf{z}_k, f_k)_{k=1}^{\lambda}$

4.3.4 Modified Approximate Ranking Procedure (ARP)

The original approximate ranking procedure proposed by Runarsson [75] has been presented in Algorithm 4.1. In the iteration of the original approximate ranking procedure, one individual is selected to be re-evaluated by original fitness function, and then the

surrogate model is updated and the parent set is selected based on the updated surrogate. This loop is stop until the parent set does not change or all the individuals in the generation have been re-evaluated. However, when the population size λ is large, which is often required in solving difficult multimodal problems or high dimensional problems, the amount of information added in one iteration may result in insignificant changes even of a surrogate model with bad ranking predictions [76]. To overcome this deficiency, we make some modifications for the original approximate ranking procedure.

Algorithm 4.5 Modified Approximate Ranking Procedure (ARP)

1: **given:** $\left(\mathbf{z}_k, \mathbf{x}_k\right)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, t, \mathcal{A}, f\left(\mathbf{x}\right), C\left(\mathbf{x}\right)$

2: approximate: $\mathcal{D}_{\mathrm{T}} \leftarrow \text{training\_set\_selection}\left(\mathcal{A}\right)$, $\hat{f} \leftarrow \text{model\_training}\left(\mathcal{D}_{\mathrm{T}}\right)$ and predict $f_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$

3: rank and determine the parent set $\mathcal{P}_1 = \left\{\mathbf{x}_{i:\lambda}\right\}_{i=1}^{\mu}$ where $\hat{f}\left(\mathbf{x}_{1:\lambda}\right) \le \hat{f}\left(\mathbf{x}_{2:\lambda}\right) \le \cdots \le \hat{f}\left(\mathbf{x}_{\lambda:\lambda}\right)$

4: select the n_{init} best individuals based on metric $\left(C\left(\mathbf{x}_k\right)\right)_{k=1}^{\lambda}$ computed by model $\hat{f}$

5: evaluate the n_{init} selected individuals by $f\left(\mathbf{x}\right)$ and add to the set $\mathcal{A}$, $t \leftarrow t + n_{\text{init}}$

6: **for** $m = 2 : \left(\lambda - n_{\text{init}}\right)/n_{\mathrm{b}}$ **do**

7: approximate: $\mathcal{D}_{\mathrm{T}} \leftarrow \text{training\_set\_selection}\left(\mathcal{A}\right)$, $\hat{f} \leftarrow \text{model\_training}\left(\mathcal{D}_{\mathrm{T}}\right)$ and predict $f_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$

8: determine the parent set $\mathcal{P}_m = \left\{\mathbf{x}_{i:\lambda}\right\}_{i=1}^{\mu}$ where $\hat{f}\left(\mathbf{x}_{1:\lambda}\right) \le \hat{f}\left(\mathbf{x}_{2:\lambda}\right) \le \cdots \le \hat{f}\left(\mathbf{x}_{\lambda:\lambda}\right)$

9: **if** $\mathcal{P}_{m-1} \neq \mathcal{P}_m$ **then** (the parent set has changed)

10: select n_{b} best individuals based on metric $\left(C\left(\mathbf{x}_k\right)\right)_{k=1}^{\lambda}$ computed by model $\hat{f}$

11: evaluate the n_{b} selected individuals by $f\left(\mathbf{x}\right)$ and add to the set $\mathcal{A}$, $t \leftarrow t + n_{\mathrm{b}}$

12: **else** (parent set remains unchanged)

13: **break** (exit for loop)

14: **end if**

15: **end for**

16: **output:** $t,\ \mathcal{A},\ \left(\mathbf{x}_k, \mathbf{z}_k, f_k\right)_{k=1}^{\lambda}$

Firstly, instead of selecting only one individual for re-evaluation and updating the surrogate before entering the iteration loop of approximate ranking procedure, we select $n_{\text{init}} = \max\left(1, \lfloor 0.3\lambda \rfloor\right)$ individuals to be re-evaluated by $f\left(\mathbf{x}\right)$ for updating the model. Obviously, more information would be added through this change. Secondly, in the iteration loop of approximate ranking procedure, we use a batch size $n_{\mathrm{b}} = \max\left(1, \lfloor \lambda/10 \rfloor\right)$ which is

proportional to λ, which is similar to that in [76]. Furthermore, since Kriging model is used, the metric (including Mean, SD, SLB, POI and EI) can be adopted in selecting individuals for re-evaluation and updating the model. With the aid of metric, the individuals with which the model can be significantly improved can be selected for re-evaluation and added to the training data set. Therefore, with above modifications, the efficiency of approximate ranking procedure could be enhanced. The modified approximate ranking procedure is expressed by Algorithm 4.5.

4.3.5 Generation-based Evolution Control

For generation-based evolution control, both fixed and adaptive control frequency can be used. In fixed generation-based evolution control, the control frequency is fixed, i.e., generation control is carried out once in a fixed number of generations. While, for adaptive generation-based evolution control, the frequency of generation control is adaptive, usually depend on the fidelity (quality) of the surrogate model. An appropriate generation control frequency can ensure the evolutionary computation converge to correct optimum and, at the same time, reduce the computational cost as much as possible.

In [74], generation control was performed in every third generation in the comparative studies of surrogate-assisted ES with pre-selection strategy and generation-based control. Jin et al. [70, 85] proposed a strategy for adapting the generation control frequency based on the fidelity of the surrogate model. This adapting strategy for control frequency is presented below.

It is intuitive that the higher the fidelity of the surrogate model is, the more often the fitness evaluation can be made using the surrogate model [70]. Because it is very difficult to estimate the global fidelity of the surrogate model, the authors proposed to use a local estimation of the model fidelity. Since the evolution strategy generally proceeds with small steps, the current error can be used to estimate the local fidelity of the model and then to determine the frequency at which the original fitness function is used and the surrogate model is updated. Consider there are p generations within an evolution control cycle, and η

generations need to be controlled. The η, which indicates the control frequency, is adapted by

$$\eta(k+1)=\eta_{\min}+\left\lfloor \frac{E(k)}{E_{\max}} \right\rfloor(\eta_{\max}-\eta_{\min}), \tag{4.22}$$

where $\lfloor x \rfloor$ denotes the largest integer that is smaller than x, $\eta_{\max}$ is the maximal value of η, $\eta_{\max} \le p$, and the minimum $\eta_{\min}$ usually equals 1 so that the information on the model fidelity is always available, $E_{\max}$ is the allowed maximal model error and $E(k)$ is the current model error estimation, k denotes the k-th evolution control cycle of p generations. The current model error, which is estimated before the next control cycle, is measured as follows:

$$E(k)=\sqrt{\frac{1}{\eta\lambda}\sum_{i=1}^{\eta\lambda}\left(f(\mathbf{x}_i)-\hat{f}(\mathbf{x}_i)\right)^2} \tag{4.23}$$

where λ is the population size of ES, $\mathbf{x}_i, i=1,2,\cdots,\eta\lambda$ are $\eta\lambda$ individuals that are evaluated using original fitness function in current control cycle. Apparently, $E(k)$ is the mean square error of $\eta\lambda$ evaluated points. To make sure that the information on the model fidelity is always available, at least one generation should be controlled within one evolution control cycle. Besides, since the model fidelity is estimated locally based on the error information from the last cycle, p should not be too large.

4.3.5.1 Proposed Adaptive Generation-based Control (FGC) based on Model Quality

In previous sections, the model quality has been used to control the model impact of pre-selection strategy. In this section, it is proposed to use the model quality to determine whether next generation is control or not. The quality of the surrogate model Q is estimated in every controlled generation, if $Q \ge Q^{\mathrm{TS}}$ (Q^{TS} is the critical value or threshold of model quality), the next generation is evaluated by surrogate model; otherwise, the next generation is controlled and evaluated by original fitness function. The model quality measurements can be the proposed selection-based model quality measure using recombination weights Q_w, selection-based model quality measure $Q_{\text{selection}}$ proposed by Ulmer et al. [84], and the rank correlation coefficient ρ_{rank}. The threshold of model quality measurements have been given in Table 4.1.

4.4 Kriging-Assisted CMA-ES (KA-CMA-ES) Algorithms

In this section, these issues described in previous section including training set selection, pre-selection with model impact control (CPS), fixed individual-based control using metrics (FIC), mixed individual-based control (MIC), modified approximate ranking procedure (ARP) and adaptive generation-based control (AGC) are incorporated into CMA-ES to compose several different KA-CMA-ES algorithms. In training and using the surrogate model, variable transformation of correlated variables is performed firstly. Before the evolution loop of KA-CMA-ES, a number of solutions (candidates or points) are initially sampled such that surrogate model can be built at the beginning of evolution loop. At the same time, the best candidate in initial sampling are chosen as the start point for evolution loop in KA-CMA-ES rather than set the start point randomly. All these are described in the remaining part of this section.

4.4.1 Variable Transformation for Model Learning and Prediction

Recently, the covariance matrix information has been used in model training so that the correlations between variables are taken into account during model learning. In lmm-CMS-ES (local-meta-model CMA-ES) [76] and its relevant studies [77, 86], the covariance matrix $\mathbf{C}$ that is adapted in CMA-ES was used as a metric in the calculation of distance between two variable vectors $d\left(\mathbf{x}_i,\mathbf{x}_j\right)=\sqrt{\left(\mathbf{x}_i-\mathbf{x}_j\right)^{\mathrm{T}}\mathbf{C}^{-1}\left(\mathbf{x}_i-\mathbf{x}_j\right)}$, where $\mathbf{x}_i\in\mathbb{R}^d$ and $\mathbf{x}_j\in\mathbb{R}^d$ are two variable vectors (search points, candidates or individuals), and $\mathbf{C}\in\mathbb{R}^{d\times d}$ is the current covariance matrix of the CMA-ES. This covariance matrix based distance takes into account the correlations of the data.

In this work, a transformation of variables which transforms correlated variables into uncorrelated is suggested to be performed before the model training and prediction. The CMA-ES, which uses general (or correlated) mutation operator, is considered. In each generation of the CMA-ES, λ offspring are generated by mutation $\mathbf{x}_k=\mathbf{m}^{(g)}+\sigma^{(g)}\mathcal{N}_d\left(\mathbf{0},\mathbf{C}^{(g)}\right),\ k=1,\cdots,\lambda$. In other words, the λ offspring are from MVN distribution with mean vector $\boldsymbol{\mu}=\mathbf{m}^{(g)}$ and covariance matrix $\boldsymbol{\Sigma}=\left(\sigma^{(g)}\right)^2\mathbf{C}^{(g)}$, i.e.,

$\mathbf{x}_k \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \mathcal{N}_d\left(\mathbf{m}^{(g)}, \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right)$. Thus, for the original variable vector $\mathbf{x}_i \in \mathbb{R}^d$, the following transformation T is used

$$\mathbf{x}_i' = \mathrm{T}(\mathbf{x}_i) = \boldsymbol{\Sigma}^{-1/2}(\mathbf{x}_i - \boldsymbol{\mu}) = \left[\left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right]^{-1/2}\left(\mathbf{x}_i - \mathbf{m}^{(g)}\right), \tag{4.24}$$

where $\mathbf{x}_i' \in \mathbb{R}^d$ is the associated vector of $\mathbf{x}_i \in \mathbb{R}^d$ obtained by the defined transformation. This transformation uses the inverse of the covariance matrix of MVN distribution and has the effects of (1) standardizing all variables in the vector to the same variance and (2) eliminating correlations between variables [35]. It is worth mentioning that standardization of data set is a common requirement for many machine learning implementations. With transformation in Equation (4.24), the Mahalanobis distance in Equation (4.7) can be rewritten as

$$\Delta(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})} = \sqrt{(\mathbf{x}_i')^{\mathrm{T}} \mathbf{x}_i'} = \sqrt{(\mathrm{T}(\mathbf{x}_i))^{\mathrm{T}} \mathrm{T}(\mathbf{x}_i)}. \tag{4.25}$$

This shows that the Mahalanobis distance between original variable vectors can be viewed as the Euclidean distance on transformed vectors.

Therefore, each time before training a surrogate model and making predictions by the model, we firstly transform all the input vectors of previously evaluated points and the current λ offspring individuals using Equation (4.24). Then, the training set is selected by using the *k*-Nearest Neighbor Points based on Mahalanobis Distance, the confidence interval of MVN or other methods. That is to say, model learning and prediction are performed on transformed variables.

4.4.2 Initial Sampling and Informed Start Point

For surrogate-assisted ES, a set of evaluated points is indispensable before the starting of surrogate assistance. Usually, the surrogate model is trained at the beginning with a randomly created initial population (or initial sampling), which can be found in [25, 72–74]. In this work, a different way is used to create initial population for constructing the initial surrogate mode. As stated in Chapter 3, space-filling design is appropriate to represent all portions of the design space. Thus, in this work, the initial sampling is created by the space-filling Latin hypercube designs described in Section 3.2.2 rather than by randomly sampling.

In other words, before the evolution loop of KA-CMA-ES, an initial design of experiments (DOE) is carried out by using Latin Hypercube Designs (LHD) based on maximin distance criterion. As for the size of the DOE, n_{DOE}, the $n_{\mathrm{DOE}} = 10D$ rule of thumb which was studied in [87] is adopted. Specifically, $n_{\mathrm{DOE}} = 10D$ (D is the dimension of the problem) sampling points are generated by LHD. However, in KA-CMA-ES using pre-selection, the size of pre-selection population is set as $\lambda_{\mathrm{Pre}} = 2\lambda$. In order to runs pre-selection after the initial sampling, n_{DOE} should not smaller than λ_{Pre}. Thus, we correspondingly set $n_{\mathrm{DOE}} = \max(10D, 2\lambda)$ in all the KA-CMA-ES algorithms which are expressed and investigated subsequently.

It is apparent that smaller number of iterations are needed to reach the optimum when the start point $\mathbf{m}^{(0)}$ for the evolution loop is near the location of the optimum, and vice versa. However, for black-box problem, usually no prior knowledge about the global optimum is available before we solve the optimization problems. Thus, the start point $\mathbf{m}^{(0)} \in \mathbb{R}^d$ is generally initialized uniformly and randomly within the search space. Fortunately, information can be extracted from the samples of initial sampling (DOE). Therefore, we proposed to choose the best point in initial sampling (the point with minimum fitness function value) as the start point for subsequent evolution loop, after the evaluation of the n_{DOE} points in DOE. This strategy for start point selection is referred to as informed start point in this work. In this work, the initial sampling and informed start point are performed before the evolution loop for all surrogate-assisted evolution algorithms.

4.4.3 KA-CMA-ES using Pre-Selection

This subsection dedicates to the KA-CMA-ES using pre-selection, where pre-selection without model impact control (PS) and pre-selection with model impact control (CPS) are incorporated into CMA-ES. The standard $(\mu/\mu_w, \lambda)$-CMA-ES, which has already presented in Section 2.3.3, is not repeated here. The KA-CMA-ES using pre-selection can be easily developed by introducing initial sampling and informed start point before the evolution loop, and then runs the pre-selection procedure before the evaluation of offspring population

(using original fitness function f) in the standard evolution loop of the $(\mu/\mu_w, \lambda)$-CMA-ES algorithm.

Algorithm 4.6 The KA-CMA-ES using Pre-Selection

1: **given:** strategy parameters of CMA-ES, pre-selection method (PS or CPS) and parameters (λ_{Pre} for PS, $\lambda_{\text{Pre}}^{(0)}$, $\delta_{\lambda_{\text{Pre}}}$ and model quality measure Q for CPS).

2: **initialize** $\sigma^{(0)} > 0, \mathbf{p}_{\text{c}}^{(0)} = \mathbf{0}, \mathbf{p}_{\sigma}^{(0)} = \mathbf{0}, \mathbf{C}^{(0)} = \mathbf{I}, \mathcal{A}=\emptyset, \mathcal{D}_{\text{T}} = \emptyset, g \leftarrow 0, t \leftarrow 0$

3: initial sampling: $\mathbf{X}_{\text{DOE}}^{\text{T}} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\text{DOE}}}\right]^{\text{T}} \leftarrow \text{lhsdesign}\left(n_{\text{DOE}}, d\right)$

4: evaluate initial samples: $f_k = f\left(\mathbf{x}_k\right), \mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_k, f_k\right)$ where $\mathbf{x}_k \in \mathbf{X}_{\text{DOE}}, k = 1, \cdots, n_{\text{DOE}}$

5: informed start point: $\mathbf{m}^{(0)} = \underset{\mathbf{x}_k \in \mathbf{X}_{\text{DOE}}^{\text{T}}}{\arg\min}\ f_k$

6: **repeat**

7: training set selection: $\mathcal{D}_{\text{T}} \leftarrow \text{training\_set\_selection}\left(\mathcal{A}\right)$

8: model training: $\hat{f} \leftarrow \text{model\_training}\left(\mathcal{D}_{\text{T}}\right)$

9: **if** PS is used **then**

10: $\left(\mathbf{x}_k, \mathbf{z}_k, \hat{f}_k\right)_{k=1}^{\lambda} \leftarrow \text{pre-selection\_procedure}\left(\mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \mathcal{A}, \lambda_{\text{Pre}}\right)$
//pre-selection without model impact control

11: **elseif** CPS is used **then**

12: $\left(\mathbf{x}_k, \mathbf{z}_k, \hat{f}_k\right)_{k=1}^{\lambda} \leftarrow \text{pre-selection\_procedure}\left(\mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \mathcal{A}, \lambda_{\text{Pre}}^{(g)}\right)$
//pre-selection with model impact control

13: **end if**

14: **for** $k = 1, \cdots, \lambda$ **do** //evaluate by original fitness function

15: $f_k = f\left(\mathbf{x}_k\right)$

16: $\mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_k, f_k\right)$

17: $t \leftarrow t + 1$

18: **end do**

19: **if** CPS is used **then**

20: model quality estimate: $Q^{(g)} \leftarrow \text{model\_qualtiy\_estimate}\left(\left\{\mathbf{x}_k, \hat{f}_k\right\}_{k=1}^{\lambda}, \left\{\mathbf{x}_k, f_k\right\}_{k=1}^{\lambda}\right)$

21: $\lambda_{\text{Pre}}^{(g+1)} = \begin{cases} \lambda_{\text{Pre}}^{(g)} + \dfrac{Q^{(g)} - Q^{\text{TS}}}{Q^{\max} - Q^{\text{TS}}}\delta_{\text{Pre}}, & \text{if } Q^{(g)} > Q^{\text{TS}} \\ \lambda_{\text{Pre}}^{(g)} - \dfrac{Q^{\text{TS}} - Q^{(g)}}{Q^{\text{TS}} - Q^{\min}}\delta_{\text{Pre}}, & \text{otherwise} \end{cases}$ //update λ_{Pre}

22: **end if**

23: update $\mathbf{m}^{(g+1)}, \sigma^{(g+1)}, \mathbf{C}^{(g+1)}$ (the same as standard CMA-ES algorithm Line 9~13)

24: $g \leftarrow g + 1$

25: **until** termination criterion is fulfilled

In each evolution iteration, the pre-selection procedure is firstly performed. After the pre-selection step, the evaluation, selection and recombination are performed on the λ pre-screened offspring in the same way as in standard CMA-ES. If there is no model impact control in pre-selection (PS), the size of pre-selection population λ_{Pre} is set by user and keeps constant during the computation. For pre-selection with model impact control (CPS), the model quality Q (including Q_w, $Q_{\text{selection}}$ and ρ_{rank}) need to be estimated after the evaluation of λ offspring by original fitness function, and then the size of pre-selection population λ_{Pre} is updated according to Equation (4.16). The pseudocode of KA-CMA-ES using Pre-Selection without and with model impact control is given in Algorithm 4.6. This algorithm contains two methods: KA-CMA-ES using PS, which presents Kriging-assisted CMA-ES using pre-selection without model impact control, and KA-CMA-ES using CPS,which stands for Kriging-assisted CMA-ES using pre-selection with model impact control. The pre-selection procedure used in Line 10 and 12 has been given in Algorithm 4.2.

4.4.4 KA-CMA-ES using Individual-based Evolution Control

For Kriging-Assisted CMA-ES using individual-based evolution control, fixed individual-based control (FIC) using metric and our proposed mixed individual-based control (MIC) can be used. In fixed individual-based control (FIC), metric is used for selecting the most promising individuals for re-evaluation. The details about metrics (Mean, SD, SLB, POI and EI) for FIC have been discussed in Section 4.3.3.1. In mixed individual-based control (MIC), two metrics, i.e., the Mean (mean of prediction) and SLB (the statistical lower bound), are provided. The pseudocode of KA-CMA-ES using individual-based control, including FIC and MIC, is presented in Algorithm 4.7.

The initial sampling and informed start point are firstly introduced in KA-CMA-ES using individual-based control (Line 3~5), before the evolution loop. In each evolution iteration, λ individuals (offspring) are generated by mutation operator (Line 7~10). Then, the fixed individual-based control (FIC) or mixed individual-based control (MIC) is performed

(Line 13~17). With the results of FIC or MIC, the distribution mean, step-size and covariance matrix can be updated (Line 18) the same as the standard CMA-ES.

Algorithm 4.7 The KA-CMA-ES using Individual-based Control

1: **given:** strategy parameters of CMA-ES, individual-based control method (FIC or MIC) and parameters (η for FIC, $\eta_{\min}$, $\eta_{\max}$ for MIC), and metric $C(\mathbf{x})$ used in individual control.

2: **initialize** $\sigma^{(0)} > 0, \mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{0}, \mathbf{p}_{\sigma}^{(0)} = \mathbf{0}, \mathbf{C}^{(0)} = \mathbf{I}, \mathcal{A}=\varnothing, \mathcal{D}_{\mathrm{T}} = \varnothing, g \leftarrow 0, t \leftarrow 0$

3: initial sampling: $\mathbf{X}_{\mathrm{DOE}}^{\mathrm{T}} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\mathrm{DOE}}}\right]^{\mathrm{T}} \leftarrow \mathrm{lhsdesign}\left(n_{\mathrm{DOE}}, d\right)$

4: evaluate initial samples: $f_k = f\left(\mathbf{x}_k\right), \mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_k, f_k\right)$ where $\mathbf{x}_k \in \mathbf{X}_{\mathrm{DOE}}, k = 1, \cdots, n_{\mathrm{DOE}}$

5: informed start point: $\mathbf{m}^{(0)} = \arg\min_{\mathbf{x}_k \in \mathbf{X}_{\mathrm{DOE}}^{\mathrm{T}}} f_k$

6: **repeat**

7: **for** $k = 1, \cdots, \lambda$ **do** //generate λ offspring

8: $\mathbf{z}_k = \mathcal{N}_d\left(\mathbf{0}, \mathbf{I}\right)$ //i.i.d. for each $\mathbf{z}_k$

9: $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{1/2} \mathbf{z}_k$ //mutation

10: **end for**

11: training set selection: $\mathcal{D}_{\mathrm{T}} \leftarrow \mathrm{training\_set\_selection}\left(\mathcal{A}\right)$

12: model training: $\hat{f} \leftarrow \mathrm{model\_training}\left(\mathcal{D}_{\mathrm{T}}\right)$

13: **if** FIC is used **then**

14: $t, \mathcal{A}, \left(\mathbf{x}_k, \mathbf{z}_k, f_k\right)_{k=1}^{\lambda} \leftarrow \mathrm{fixed\_individual\_control}\begin{pmatrix}\left(\mathbf{z}_k, \mathbf{x}_k\right)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \\ t, \mathcal{A}, f\left(\mathbf{x}\right), \eta, C\left(\mathbf{x}\right)\end{pmatrix}$

15: **elseif** MIC is used **then**

16: $t, \mathcal{A}, \left(\mathbf{x}_k, \mathbf{z}_k, f_k\right)_{k=1}^{\lambda} \leftarrow \mathrm{mixed\_individual\_control}\begin{pmatrix}\left(\mathbf{z}_k, \mathbf{x}_k\right)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \\ t, \mathcal{A}, f\left(\mathbf{x}\right), \eta_{\min}, \eta_{\max}, C\left(\mathbf{x}\right)\end{pmatrix}$

17: **end if**

18: update $\mathbf{m}^{(g+1)}, \sigma^{(g+1)}, \mathbf{C}^{(g+1)}$ (the same as standard CMA-ES algorithm Line 9~13)

19: $g \leftarrow g + 1$

20: **until** termination criterion is fulfilled

4.4.5 KA-CMA-ES using Approximate Ranking Procedure (ARP)

Previously described modified approximate ranking procedure, which has been illustrated in Algorithm 4.5, is embedded into CMA-ES. The pseudocode of KA-CMA-ES

using approximate ranking procedure (ARP) is given in Algorithm 4.8. The initial sampling and informed start point are firstly performed (Line 3~5). In each evolution loop, the approximate ranking procedure is called after λ individuals have been generated.

Algorithm 4.8 The KA-CMA-ES using Approximate Ranking Procedure (ARP)

1: **given:** strategy parameters of CMA-ES, parameters for approximate ranking procedure ($n_{\text{init}}, n_{\text{b}}$ and metric $C(\mathbf{x})$).
2: **initialize** $\sigma^{(0)} > 0, \mathbf{p}_{\text{c}}^{(0)} = \mathbf{0}, \mathbf{p}_{\sigma}^{(0)} = \mathbf{0}, \mathbf{C}^{(0)} = \mathbf{I}, \mathcal{A}=\varnothing, \mathcal{D}_{\text{T}} = \varnothing, g \leftarrow 0, t \leftarrow 0$
3: initial sampling: $\mathbf{X}_{\text{DOE}}^{\text{T}} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\text{DOE}}}\right]^{\text{T}} \leftarrow \text{lhsdesign}\left(n_{\text{DOE}}, d\right)$
4: evaluate initial samples: $f_k = f\left(\mathbf{x}_k\right), \mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_k, f_k\right)$ where $\mathbf{x}_k \in \mathbf{X}_{\text{DOE}}, k = 1, \cdots, n_{\text{DOE}}$
5: informed start point: $\mathbf{m}^{(0)} = \underset{\mathbf{x}_k \in \mathbf{X}_{\text{DOE}}^{\text{T}}}{\arg\min}\ f_k$
6: **repeat**
7: **for** $k = 1, \cdots, \lambda$ **do** //standard mutation and evaluation of CMA-ES
8: $\mathbf{z}_k = \mathcal{N}_d\left(\mathbf{0}, \mathbf{I}\right)$ //i.i.d. for each $\mathbf{z}_k$
9: $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{1/2} \mathbf{z}_k$
10: **end for**
11: $t, \mathcal{A}, \left(\mathbf{x}_k, \mathbf{z}_k, f_k\right)_{k=1}^{\lambda} \leftarrow \text{approximate\_ranking\_procedure}\left(\left(\mathbf{z}_k, \mathbf{x}_k\right)_{k=1}^{\lambda}, \mathbf{m}^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, t, \mathcal{A}, f\left(\mathbf{x}\right), C\left(\mathbf{x}\right)\right)$
12: update $\mathbf{m}^{(g+1)}, \sigma^{(g+1)}, \mathbf{C}^{(g+1)}$ (the same as standard CMA-ES algorithm Line 9~13)
13: $g \leftarrow g + 1$
14: **until** termination criterion is fulfilled

4.4.6 KA-CMA-ES using Adaptive Generation-based Control (AGC)

The CMA-ES is further combined with the adaptive generation-based control (AGC) strategy described in Section 4.3.5, in which the model quality is used to control the frequency of generation control. Specifically, in adaptive generation-based control, if the model quality $Q^{(g)}$ of current controlled generation exceeds the threshold value Q^{TS}, next generation is evaluated using the surrogate model; otherwise, next generation is controlled. The pseudocode of KA-CMA-ES using adaptive generation-based control (AGC) is expressed in Algorithm 4.9. The model quality measures Q_w, $Q_{\text{selection}}$ and ρ_{rank} can be used in AGC. In Algorithm 4.9, $I_{\text{control}}^{(g)}$ indicates the generation g is controlled or not. If $I_{\text{control}}^{(g)} = 1$,

generation g is controlled (evaluated by the original fitness function), otherwise, generation g is evaluated by the surrogate model.

Algorithm 4.9 The KA-CMA-ES using Adaptive Generation-based Control (AGC)

1: **given:** strategy parameters of CMA-ES, model quality Q for AGC
2: **initialize** $\sigma^{(0)} > 0, \mathbf{p}_{\mathrm{c}}^{(0)} = \mathbf{0}, \mathbf{p}_{\sigma}^{(0)} = \mathbf{0}, \mathbf{C}^{(0)} = \mathbf{I}, \mathcal{A}=\varnothing, \mathcal{D}_{\mathrm{T}} = \varnothing, I_{\mathrm{control}}^{(0)} = 1, g \leftarrow 0, t \leftarrow 0$
3: initial sampling: $\mathbf{X}_{\mathrm{DOE}}^{\mathrm{T}} = \left[\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\mathrm{DOE}}}\right]^{\mathrm{T}} \leftarrow \mathrm{lhsdesign}\left(n_{\mathrm{DOE}}, d\right)$
4: evaluate initial samples: $f_k = f\left(\mathbf{x}_k\right), \mathcal{A} \leftarrow \mathcal{A} \cup \left(\mathbf{x}_k, f_k\right)$ where $\mathbf{x}_k \in \mathbf{X}_{\mathrm{DOE}}, k = 1, \cdots, n_{\mathrm{DOE}}$
5: informed start point: $\mathbf{m}^{(0)} = \arg\min_{\mathbf{x}_k \in \mathbf{X}_{\mathrm{DOE}}^{\mathrm{T}}} f_k$
6: **repeat**
7: **for** $k = 1, \cdots, \lambda$ **do** //standard mutation and evaluation of CMA-ES
8: $\mathbf{z}_k = \mathcal{N}_d\left(\mathbf{0}, \mathbf{I}\right)$ //i.i.d. for each $\mathbf{z}_k$
9: $\mathbf{x}_k = \mathbf{m}^{(g)} + \sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{1/2} \mathbf{z}_k$
10: **end for**
11: training set selection: $\mathcal{D}_{\mathrm{T}} \leftarrow \mathrm{training\_set\_selection}\left(\mathcal{A}\right)$
12: model training: $\hat{f} \leftarrow \mathrm{model\_training}\left(\mathcal{D}_{\mathrm{T}}\right)$
13: prediction: $\hat{f}_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$
14: **If** $I_{\mathrm{control}}^{(g)}$ is equal 1 **then**
15: evaluation: $f_k = f\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$
16: model quality estimate: $Q^{(g)} \leftarrow \mathrm{model\_qualtiy\_estimate}\left(\left\{\mathbf{x}_k, \hat{f}_k\right\}_{k=1}^{\lambda}, \left\{\mathbf{x}_k, f_k\right\}_{k=1}^{\lambda}\right)$
17: **if** $Q^{(g)} \geq Q^{\mathrm{TS}}$ **then**
18: $I_{\mathrm{control}}^{(g+1)} = 0$ //mark next generation as model-evaluated
19: **else**
20: $I_{\mathrm{control}}^{(g+1)} = 1$ //mark next generation as controlled
21: **end if**
22: **esleif** $I_{\mathrm{control}}^{(g)}$ is equal 0 **then**
23: $f_k = \hat{f}\left(\mathbf{x}_k\right), k = 1, \cdots, \lambda$
24: $I_{\mathrm{control}}^{(g+1)} = 1$ //mark next generation as controlled
25: **end if**
26: update $\mathbf{m}^{(g+1)}, \sigma^{(g+1)}, \mathbf{C}^{(g+1)}$ (the same as standard CMA-ES algorithm Line 9~13)
27: $g \leftarrow g + 1$
28: **until** termination criterion is fulfilled

4.5 Experimental Studies

The validation and the performance evaluation of optimization algorithms is commonly carried out by using a chosen set of benchmarks or test functions. In this section, the proposed Kriging-assisted CMA-ES algorithms in Section 4.4 are validated using a set of test functions and their performance are evaluated and analyzed.

4.5.1 Experimental Setup

4.5.1.1 Test Functions

Test functions are important in performance validation and comparison of optimization algorithms. In order to comprehensively evaluate an algorithm, the used set of test functions should includes enough functions with different characteristics, such as continuous, discontinuous, unimodal, multi-modal, separable, non-separable. In this work, we focus on continuous optimization problems. In the experimental studies of proposed Kriging-Assisted CMA-ES (KA-CMA-ES) algorithms, a set of 12 continuous benchmark functions is carefully selected from [88–90].

All the test functions are minimization problems defined as follows :

$$\begin{aligned} &\text{minimize } f(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, \cdots, x_D]^{\mathrm{T}} \\ &\text{subject to } \mathbf{x} \in S = [\mathbf{x}_{\mathrm{LB}}, \mathbf{x}_{\mathrm{UB}}] \end{aligned} \tag{4.26}$$

where $f(\mathbf{x})$ is objective or fitness function, D is the dimension of the problem (the number of parameters or variables), and S is the search space (search domain) defined by the lower bounds $\mathbf{x}_{\mathrm{LB}} \in \mathbb{R}^D$ and upper bounds $\mathbf{x}_{\mathrm{UB}} \in \mathbb{R}^D$. For handling the box constraints, the re-sampling method, i.e. re-sampling any infeasible solution $\mathbf{x}$ until it becomes feasible, is adopted.

The name, expression and the search space of 12 test functions are listed in Table 4.2, in which $f_1 \sim f_7$ are unimodal functions and $f_8 \sim f_{12}$ are multimodal problems. All these test functions have the global optimum $f(\mathbf{x}^*) = 0$, and the global optimum are located at $\mathbf{x}^* = [0, 0, \cdots, 0]^{\mathrm{T}}$ except Rosenbrock function, whose optimum is at $\mathbf{x}^* = [1, 1, \cdots, 1]^{\mathrm{T}}$. In our

studies, for $f_1 \sim f_5$, the dimensionality of search space are $D = 2,5,10$ and 20; for $f_6 \sim f_{11}$, we take $D = 2,5$ and 10, and f_{12} has dimension $D = 2$ and 5. Each test function with each dimension can be considered as an optimization problem. Totally, there are 40 test problems.

Table 4.2 Test functions for experimental studies

Name	Function	Search Space
Sphere	$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-5,5]^D$
Bent Cigar	$f_2(\mathbf{x}) = x_1^2 + 10^4 \sum_{i=2}^{D} x_i^2$	$[-100,100]^D$
Sum Squares	$f_3(\mathbf{x}) = \sum_{i=1}^{D} \left(i \cdot x_i^2\right)$	$[-10,10]^D$
Schwefel 1.2	$f_4(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100,100]^D$
Powell Sum	$f_5(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	$[-1,1]^D$
Schwefel Absolute	$f_6(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-100,100]^D$
Rosenbrock	$f_7(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$	$[-5,5]^D$
Ackley	$f_8(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi x_i\right) \right) + 20 + \exp(1)$	$[-32,32]^D$
Levy	$f_9(\mathbf{x}) = \sin^2(\pi z_1) + \sum_{i=1}^{D-1} \left(z_i - 1\right)^2 \left[1 + 10\sin^2\left(\pi z_i + 1\right)\right] + \left(z_D - 1\right)^2 \left[1 + \sin^2\left(2\pi z_D\right)\right]$	$[-10,10]^D$
Weierstrass	$f_{10}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right) \right] \right) - D\sum_{k=0}^{k\max} \left[a^k \cos\left(2\pi b^k \cdot 0.5\right) \right]$ where $a = 0.5, b = 3, k\max = 20$	$[-0.5,0.5]^D$
Bohachevsky	$f_{11}(\mathbf{x}) = \sum_{i=1}^{D-1} \left[x_i^2 + 2x_{i+1}^2 - 0.3\cos\left(3\pi x_i\right) - 0.4\cos\left(4\pi x_{i+1}\right) + 0.7 \right]$	$[-15,15]^D$
Rastrigin	$f_{12}(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left[x_i^2 - 10\cos\left(2\pi x_i\right) \right]$	$[-5,5]^D$

4.5.1.2 Experimental Setting

For each test problem, 25 independent runs are performed using the algorithm which are chosen to be investigated. The initial mean is set as: for CMA-ES, the initial mean $\mathbf{m}^{(0)}$ is uniformly sampled within the search domain of the problem (given in Table 4.2) ; for KA-

CMA-ES, the initial mean $\mathbf{m}^{(0)}$ of the evolution loop is the best candidate in initial sampling (informed start point), where $n_{\mathrm{DOE}} = \max(10D, 2\lambda)$ samples are generated by LHD within the search space. A single run of the algorithm is terminated, when the target function (fitness) value $f_{\mathrm{target}} = 10^{-10}$ is reached or one of the following termination conditions is satisfied:

maxFES: maximum number of exact fitness function evaluations, for CMA-ES, $\mathrm{maxFES} = 10^4 D$; for KA-CMA-ES, $\mathrm{maxFES} = 10^4$.

ConditionCov: the condition number of $\mathbf{C}^{(g)}$ exceeds 10^{14}.

TolFun: stop if the range of the best objective function values of the last $10 + \lceil 30D/\lambda \rceil$ generations and all function values of the recent generation is below $\mathrm{TolFun} = 10^{-10}$.

TolX: stop if all components of $\mathbf{p}_c^{(g)}$ and all square roots of diagonal components of $\mathbf{C}^{(g)}$, multiplied by $\sigma^{(g)}/\sigma^{(0)}$, are smaller than $\mathrm{TolX} = 10^{-12}$.

4.5.1.3 Evaluation Criteria

For each run of the problems, the number of exact function evaluations (FES) and current best fitness function values are recorded. All the best fitness function values should be from solutions that are evaluated using original fitness function. In other words, in KA-CMA-ES, the number of exact function evaluations (evaluating using original fitness function) and corresponding current best function values of evaluated individuals are recorded.

In experimental study, the performance of an algorithm is evaluated according to the success rate (SR), the success performance (SP) and the speedup performance (SPU). The optimization runs which reach the target objective function value f_{target} are known as successful runs. While, these runs that do not reach f_{target} are considered unsuccessful. A successful run is rated with the number of function evaluations to reach f_{target}, i.e., the FES of successful run. The success rate (SR) is the ratio of number of successful runs to total runs, which is computed as

$$\mathrm{SR} = \frac{\#\ \text{of successful runs}}{\#\ \text{of total runs}} \tag{4.27}$$

The success performance (SP) can be regarded as the expected number of FES to reach f_{target}, and is given by

$$\text{SP} = \text{mean}\left(\text{FES of successful runs}\right) \times \frac{\#\text{ of total runs}}{\#\text{ of successful runs}} \tag{4.28}$$

Based on success performance (SP), the speedup performance (SPU) of KA-CMA-ES is defined as the ratio between SP of CMA-ES and that of the investigated KA-CMA-ES algorithm, i.e.,

$$\text{SPU} = \frac{\text{SP of CMA-ES}}{\text{SP of KA-CMA-ES}} \tag{4.29}$$

The speedup performance (SPU) denotes the degree of improvement in performance (according to number of exact function evaluations) of the KA-CMA-ES algorithm. Obviously, the SPU of an algorithms is larger than one (SPU>1) means the algorithm makes improvement in success performance. On the contrary, SPU<1 indicates the algorithm performs worse than standard CMA-ES.

4.5.2 Experiments on KA-CMA-ES using Pre-Selection

4.5.2.1 Pre-Selection without Model Impact Control (PS)

Three training set selection methods, including Recently Evaluated Points, *k*-Nearest Neighbor Points to distribution mean based on Mahalanobis distance and Confidence Interval, have been investigated in KA-CMA-ES using pre-selection without model impact control (PS). The size of training set is set as $n_{\text{T}} = 2\lambda$ for Recently Evaluated Points and k-Nearest Neighbor Points methods, and $p_{\text{training}} = 99.73\%$ for Confidence Interval methods. The size of pre-selection population is set as $\lambda_{\text{Pre}} = 2\lambda$ and keeps as constant during the computation. For simplification reason, we use ‘Recently’ to represents the Recently Evaluated Points method of training set selection, ‘kNN’ to stand for the *k*-Nearest Neighbor Points to distribution mean based on Mahalanobis distance, and ‘Interval’ to signify the proposed Confidence Interval method. The goal of running KA-CMA-ES using PS (KA-CMA-ES using pre-selection without model impact control) with different training set selection methods is to investigate the performance of training set selection methods and

thus to determine which method is used in subsequent experiments of other KA-CMA-ES algorithms. Additionally, the performance of PS algorithms will be compared with the pre-selection with model impact control (CPS) which are studied in next subsection.

The experimental results of KA-CMA-ES using pre-selection without model impact control (including success rate (SR), success performance (SP) and speedup performance (SPU)), in which training set selection methods 'Recently', 'kNN' and 'Interval' are adopted, are listed in Table 4.3. In this table, PS-Recently indicates the pre-selection using Recently Evaluated Points method for training set selection, PS-kNN stands for the pre-selection where training set is selected as *k*-Nearest Neighbor Points to distribution mean based on Mahalanobis distance, and PS-Interval signifies pre-selection using the proposed Confidence Interval method as training set selection.

From Table 4.3, it can be found that, for large majority of the test problems, the speedup performance (SPU) of PS-Interval is larger than that of PS-Recently and PS-kNN, which indicates better performance of PS-Interval. Additionally, PS-Interval can get higher success rate (SR) values on difficult problems such as f_{11} and f_{12}. It can be initially concluded that the proposed Confidence Interval method for training set selection, generally, work better than 'Recently' and 'kNN' methods.

Furthermore, in order to analyze the performance of different training set selection methods, we have classified the total 40 test problems into different categories (or groups) according to their dimensionality and modality, and then evaluate the performance of PS using different training selection methods on each category and overall problems. The total test problems are grouped into: 2 dimensional problems (D=2), 5 dimensional problems (D=5), 10 dimensional problems (D=10), 20 dimensional problems (D=20), unimodal problem, and multimodal problems. For each category of problems and the overall problems, we compute the average success rate (SR) and average speedup performance (SPU). This is also used in subsequent investigation of other KA-CMA-ES algorithms. The average success rate and speedup performance are listed in Table 4.4 and plotted in radar charts (also known as spider chats or start plots) of Figure 4.3.

Table 4.3 Results of KA-CMA-ES using pre-selection without model impact control (PS)

Function	D	λ	PS-Recently			PS-kNN			PS-Interval			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	253	**1.31**	1	255	1.30	1	257	1.29	1	331
	5	8	1	718	1.24	1	708	1.25	1	705	**1.26**	1	888
	10	10	1	1535	1.16	1	1521	1.17	1	1410	**1.27**	1	1784
	20	12	1	3110	1.06	1	3090	1.07	1	2902	**1.13**	1	3292
f2	2	6	1	326	1.19	1	323	1.20	1	319	**1.21**	1	387
	5	8	1	971	**1.16**	1	998	1.13	1	1004	1.12	1	1128
	10	10	1	2632	1.12	1	2422	**1.21**	1	2509	1.17	1	2941
	20	12	0.96	6422	**1.16**	1	6742	1.11	0.96	6634	1.13	1	7470
f3	2	6	1	279	1.28	1	283	1.26	1	274	**1.30**	1	356
	5	8	1	796	1.23	1	801	1.22	1	760	**1.29**	1	980
	10	10	1	1850	1.12	1	1808	1.14	1	1661	**1.25**	1	2070
	20	12	1	4416	**0.98**	1	4223	1.02	1	4194	**1.03**	1	4327
f4	2	6	1	325	**1.28**	1	336	1.24	1	326	1.28	1	417
	5	8	1	979	1.25	1	995	1.23	1	941	**1.30**	1	1224
	10	10	1	2535	1.13	1	2530	1.13	1	2307	**1.24**	1	2858
	20	12	1	7172	1.04	1	7114	1.04	1	6825	**1.09**	1	7431
f5	2	6	1	202	**1.37**	1	219	1.26	1	216	1.28	1	277
	5	8	1	878	**1.31**	1	888	1.29	1	912	1.26	1	1149
	10	10	1	2866	1.24	0.92	3173	1.12	1	2794	**1.27**	1	3556
	20	12	0.72	12036	1.16	0.88	9843	**1.42**	0.84	10444	1.34	0.76	14014
f6	2	6	1	578	1.35	1	574	**1.36**	1	578	1.35	1	778
	5	8	1	1745	1.29	1	1699	1.33	1	1599	**1.41**	1	2253
	10	10	1	4133	1.23	1	4109	1.24	1	3709	**1.37**	1	5078
f7	2	6	1	470	1.41	1	456	1.45	1	449	**1.48**	1	663
	5	8	1	1744	1.42	1	1809	1.36	0.96	1675	**1.47**	0.96	2468
	10	10	0.88	6503	1.27	0.96	5895	1.40	0.96	5073	**1.63**	0.8	8248
f8	2	6	1	553	**1.30**	1	560	1.28	1	557	1.29	1	717
	5	8	1	1464	1.39	0.96	1513	1.35	1	1414	**1.44**	0.92	2040
	10	10	0.96	3163	1.19	1	3052	1.23	1	2810	**1.34**	0.96	3753
f9	2	12	1	390	1.25	1	381	1.28	1	376	**1.29**	1	486
	5	16	1	1141	1.30	1	1137	**1.31**	1	1137	**1.31**	0.96	1489
	10	20	1	2492	1.22	0.96	2596	1.17	1	2440	**1.25**	1	3049
f10	2	12	1	846	**1.25**	1	852	1.24	0.96	901	1.17	1	1054
	5	16	1	2354	**1.33**	1	2399	1.31	1	2362	1.33	0.96	3137
	10	20	1	5144	1.34	1	5136	1.34	1	4924	**1.40**	0.92	6893
f11	2	12	1	454	**1.31**	1	461	1.30	1	460	1.30	1	597
	5	16	0.8	1618	1.05	0.92	1405	1.21	**1**	1299	**1.31**	0.96	1700
	10	20	0.88	3152	1.41	0.76	3673	1.21	**0.92**	2949	**1.51**	0.76	4457
f12	2	50	0.92	1508	1.57	0.96	1452	**1.63**	0.92	1496	1.59	0.68	2372
	5	140	0.6	9722	1.03	0.76	8338	1.20	**0.84**	7452	**1.34**	0.8	10020

Table 4.4 Average success rate (SR) and speedup performance (SPU) of KA-CMA-ES using pre-selection without model impact control (PS).

Category	Average Success Rate (SR)				Average Speedup Performance (SPU)		
	PS-Recently	PS-kNN	PS-Interval	CMA-ES	PS-Recently	PS-kNN	PS-Interval
D=2	0.993	0.997	0.990	0.973	1.322	1.316	1.318
D=5	0.950	0.970	0.983	0.963	1.251	1.267	1.321
D=10	0.975	0.964	0.989	0.949	1.221	1.216	1.335
D=20	0.936	0.976	0.960	0.952	1.080	1.133	1.145
Unimodal	0.983	0.991	0.989	0.982	1.221	1.230	1.277
Multimodal	0.940	0.951	0.974	0.923	1.282	1.290	1.347
Overall	0.968	0.977	0.984	0.961	1.242	1.251	1.302

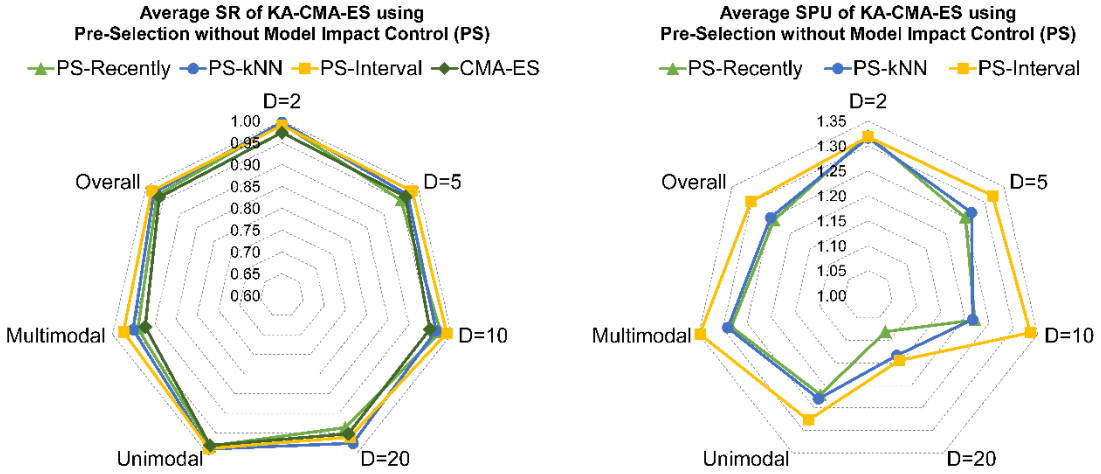

Figure 4.3 Radar charts of average success rate and speedup performance of KA-CMA-ES using pre-selection without model impact control (PS).

From the radar chart of average success rate (SR) in left of Figure 4.3 and Table 4.4, for each category of problems, the KA-CMA-ES using PS have average success rate exceeds 0.9. Specifically, the PS-kNN has the highest average success rate on problems with *D*=2 and *D*=20, and unimodal problems. PS-Interval has the highest average success rate on problems with *D*=5, *D*=10, multimodal problems, and on overall problems. The PS-Recently method has the lowest average success rate among three methods. PS-kNN and PS-Interval has higher average success rate than the standard CMA-ES on all categories of problems. In terms of speedup performance (SPU) or success performance, PS-Interval has the best success performance on all other category problems except 2 dimensional problems. Thus, it can also be found that the 'Interval' method is superior to 'Recently' and 'kNN' methods.

From the experimental study of KA-CMA-ES using pre-selection without model impact control (PS), it can be found that the proposed Confidence Interval method for training set selection in KA-CMA-ES is appropriate. Considering the success rate and success performance as a whole, it is can be concluded that the 'Interval' (Confidence Interval) method for training set selection has better performance than 'Recently' and 'kNN' methods. By all accounts, the proposed Confidence Interval method clearly superior to the commonly used 'Recently' and 'kNN' methods for training set selection. Consequently, Confidence Interval method of training set selection is adopted for training set selecting in the remaining KA-CMA-ES algorithms owing to its superiority.

4.5.2.2 Pre-Selection with Model Impact Control (CPS)

In KA-CMA-ES using pre-selection with model impact control (CPS), three model impact control approaches, in which three different model quality measures including Q_w, $Q_{\text{selection}}$ and ρ_{rank} are used, are investigated. In experiments of KA-CMA-ES using CPS, we set the initial pre-selection population size $\lambda_{\text{Pre}}^{(0)} = 2\lambda$ and the adaptation rate $\delta_{\lambda_{\text{Pre}}} = 2$.

The experiments results of KA-CMA-ES using pre-selection with model impact control (CPS), i.e., the success rate (SR), success performance (SP) and speedup performance (SPU), are listed in Table 4.5. In the table, CPS-Q_w indicates pre-selection with model impact control using Q_w as model quality measure, CPS-$Q_{\text{selection}}$ represents pre-selection with model impact control using $Q_{\text{selection}}$ as model quality measure, and CPS-ρ_{rank} signifies pre-selection with model impact control using ρ_{rank} as model quality measure.

According to the values of SP and SPU in Table 4.5, for most problems, CPS-Q_w performs better than CPS-$Q_{\text{selection}}$ and CPS-ρ_{rank}. However, the values of SR of CPS-ρ_{rank} are generally higher than that of CPS-Q_w and CPS-$Q_{\text{selection}}$. For the problems of 20 dimensional f_3 and f_4, and 2 dimensional f_{10}, the performance of CPS-ρ_{rank} significantly overweights CPS-Q_w and CPS-$Q_{\text{selection}}$. It is worth to note that CPS-ρ_{rank} and CPS-Q_w perform worsen than CMA-ES on the problems of 20 dimensional f_3 and f_4 and 2 dimensional f_{10}. Thus, it can be said that CPS-ρ_{rank} is more stable and reliable than CPS-Q_w and CPS-$Q_{\text{selection}}$, and that CPS-Q_w generally has better success performance and speedup performance.

Table 4.5 Results of KA-CMA-ES using pre-selection with model impact control (CPS).

Function	D	λ	CPS-Q_w			CPS-$Q_{\text{selection}}$			CPS-ρ_{rank}			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	206	1.61	1	209	1.58	1	202	**1.64**	1	331
	5	8	1	541	**1.64**	1	544	1.63	1	545	1.63	1	888
	10	10	1	1122	**1.59**	1	1159	1.54	1	1220	1.46	1	1784
	20	12	1	2767	1.19	1	2756	**1.19**	1	2871	1.15	1	3292
f2	2	6	1	256	**1.51**	1	264	1.47	1	258	1.50	1	387
	5	8	1	797	1.42	1	753	**1.50**	1	814	1.39	1	1128
	10	10	1	1864	1.58	1	1829	1.61	1	1782	**1.65**	1	2941
	20	12	1	3778	**1.98**	1	3974	1.88	1	4330	1.73	1	7470
f3	2	6	1	220	1.62	1	216	**1.65**	1	222	1.60	1	356
	5	8	1	589	**1.66**	1	598	1.64	1	595	1.65	1	980
	10	10	1	1342	**1.54**	1	1389	1.49	1	1468	1.41	1	2070
	20	12	1	5307	**0.82**	1	5422	**0.80**	1	4140	**1.05**	1	4327
f4	2	6	1	254	1.64	1	256	1.63	1	252	**1.65**	1	417
	5	8	1	710	**1.72**	1	719	1.70	1	723	1.69	1	1224
	10	10	1	2102	1.36	1	2084	**1.37**	1	2224	1.29	1	2858
	20	12	0.6	15281	**0.49**	0.6	14963	**0.50**	1	6754	**1.10**	1	7431
f5	2	6	1	163	**1.70**	1	165	1.68	1	163	**1.70**	1	277
	5	8	1	572	**2.01**	1	594	1.93	1	596	1.93	1	1149
	10	10	1	1588	**2.24**	1	1693	2.10	1	1712	2.08	1	3556
	20	12	1	4279	**3.28**	1	4548	3.08	1	4852	2.89	0.76	14014
f6	2	6	1	448	**1.74**	1	461	1.69	1	466	1.67	1	778
	5	8	1	1236	**1.82**	1	1291	1.75	1	1358	1.66	1	2253
	10	10	1	2881	1.76	1	2853	**1.78**	1	3305	1.54	1	5078
f7	2	6	1	308	**2.15**	1	311	2.13	1	320	2.07	1	663
	5	8	0.96	1104	**2.24**	1	1105	2.23	0.96	1222	2.02	0.96	2468
	10	10	0.96	4013	**2.06**	0.92	4444	1.86	0.96	4555	1.81	0.8	8248
f8	2	6	1	431	**1.66**	1	438	1.64	0.96	463	1.55	1	717
	5	8	1	1043	**1.96**	1	1067	1.91	1	1070	1.91	0.92	2040
	10	10	1	2096	**1.79**	1	2153	1.74	1	2364	1.59	0.96	3753
f9	2	12	1	342	1.42	1	341	**1.43**	1	344	1.41	1	486
	5	16	1	977	**1.52**	1	985	1.51	1	989	1.51	0.96	1489
	10	20	1	2046	**1.49**	0.96	2180	1.40	1	2198	1.39	1	3049
f10	2	12	0.52	1493	**0.71**	0.4	1878	**0.56**	0.92	865	**1.22**	1	1054
	5	16	1	1899	**1.65**	1	1946	1.61	1	2141	1.47	0.96	3137
	10	20	1	3946	**1.75**	1	4058	1.70	1	4717	1.46	0.92	6893
f11	2	12	1	399	**1.50**	1	404	1.48	1	416	1.44	1	597
	5	16	0.96	1134	1.50	1	1089	**1.56**	0.92	1204	1.41	0.96	1700
	10	20	0.92	2460	**1.81**	0.88	2635	1.69	0.8	2970	1.50	0.76	4457
f12	2	50	0.8	1700	1.40	0.88	1532	1.55	0.96	1417	**1.67**	0.68	2372
	5	140	0.64	9092	1.10	0.8	7376	1.36	0.84	7222	**1.39**	0.8	10020

Table 4.6 Average success rate (SR) and speedup (SPU) performance of KA-CMA-ES using pre-selection with model impact control (CPS).

Category	Average Success Rate (SR)				Average Speedup (SPU)		
	CPS-Q_w	CPS-$Q_{\text{selection}}$	CPS-ρ_{rank}	CMA-ES	CPS-Q_w	CPS-$Q_{\text{selection}}$	CPS-ρ_{rank}
D=2	0.943	0.940	0.987	0.973	1.554	1.540	1.594
D=5	0.963	0.983	0.977	0.963	1.687	1.695	1.637
D=10	0.989	0.978	0.978	0.949	1.724	1.662	1.561
D=20	0.920	0.920	1.000	0.952	1.549	1.490	1.581
Unimodal	0.982	0.982	0.997	0.982	1.706	1.669	1.651
Multimodal	0.917	0.923	0.957	0.923	1.518	1.510	1.493
Overall	0.959	0.961	0.983	0.961	1.640	1.614	1.596

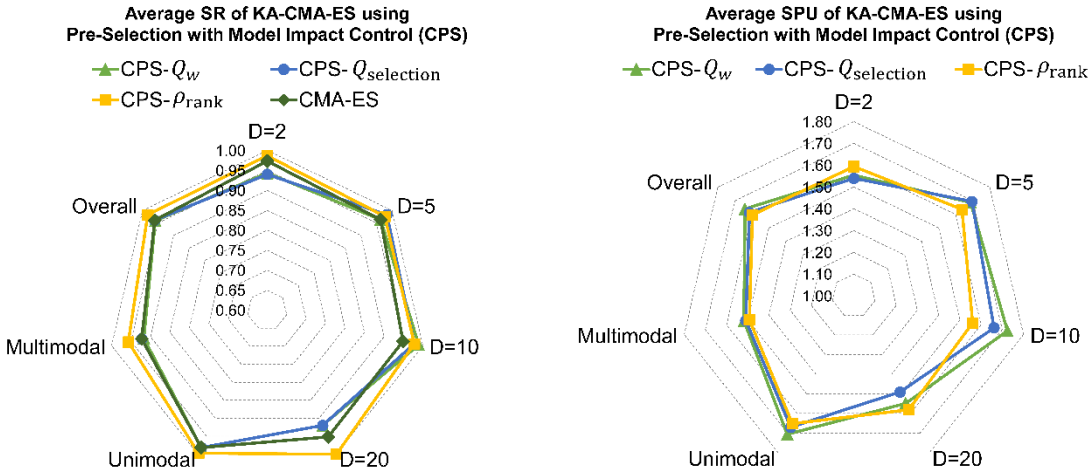

Figure 4.4 Average success rate and outperform rate of pre-selection with model impact control using different model quality measures.

The average success rate and speedup performance of CPS are listed in Table 4.6 and plotted in Figure 4.4. For problems with D=2 and D=20, unimodal problems and multimodal problems, the CPS-ρ_{rank} has the highest average success rate. The CPS-Q_w has highest average success rate on 10 dimensional problems, and CPS-$Q_{\text{selection}}$ has highest success rate on 5 dimensional problems. However, according to the speedup performance, CPS-Q_w generally outperforms CPS-$Q_{\text{selection}}$ and CPS-ρ_{rank} according to the overall average SPU.

With above results, it is hard to state which one in the three investigated CPS algorithms is outstanding. If the success performance or speedup performance is considered, CPS-Q_w is preferable. When higher success rate and the stability of algorithm are expected, CPS-ρ_{rank} is suggested. From the performance of pre-selection with model impact control using Q_w, it

can be found that the proposed model quality measure Q_w is an appropriate measurement of surrogate model quality in KA-CMA-ES.

4.5.2.3 Comparison of Pre-Selection with and without Model Impact Control

The results of KA-CMA-ES using pre-selection without model impact control using 'Interval' training set selection method (PS-Interval) and KA-CMA-ES using pre-selection with model impact control are summarized in Table 4.7. It is apparent that pre-selection with model impact control (CPS) outperforms pre-selection without model impact control (PS) for most of the test problems according to the speedup performance. Thus, model impact control is suggested in KA-CMA-ES using pre-selection.

The average success rate and speedup performance of PS-Interval, CPS-Q_w, CPS-$Q_{\text{selection}}$ and CPS-ρ_{rank} for different categories of problems are also presented in Table 4.8 and plotted in Figure 4.5. In the radar chart of speedup performance in the right of Figure 4.5, the CPS-Q_w obviously performs better than other pre-selection methods. It is clear that the pre-selection with model impact control outperforms pre-selection without model impact control. All the pre-selection methods have average success rate larger than 0.9. The CPS-ρ_{rank} generally has the highest and stable success rate for all categories of test problems, especially on 20 dimensional problems and multimodal problems. Yet the average success performance (or speedup performance) of CPS-ρ_{rank} is not so good as that of CPS-Q_w.

As discussed above, taking the success rate, speedup performance and the reliability into accounts, CPS-ρ_{rank} is preferable in all the investigated algorithms of KA-CMA-ES using pre-selection strategy. If only the success performance or speedup performance is considered, CPS-Q_w is outstanding in the investigated algorithms of pre-selection.

Table 4.7 Comparison of KA-CMA-ES using pre-selection without and with model impact control, including PS-Interval, CPS-Q_w, CPS-$Q_{\text{selection}}$ and CPS-ρ_{rank}.

Function	D	λ	PS-Interval		CPS-Q_w		CPS-$Q_{\text{selection}}$		CPS-ρ_{rank}	
			SR	SPU	SR	SPU	SR	SPU	SR	SPU
f1	2	6	1	1.29	1	1.61	1	1.58	1	**1.64**
	5	8	1	1.26	1	**1.64**	1	1.63	1	1.63
	10	10	1	1.27	1	**1.59**	1	1.54	1	1.46
	20	12	1	1.13	1	1.19	1	**1.19**	1	1.15
f2	2	6	1	1.21	1	**1.51**	1	1.47	1	1.50
	5	8	1	1.12	1	1.42	1	**1.50**	1	1.39
	10	10	1	1.17	1	1.58	1	1.61	1	**1.65**
	20	12	0.96	1.13	1	**1.98**	1	1.88	1	1.73
f3	2	6	1	1.30	1	1.62	1	**1.65**	1	1.60
	5	8	1	1.29	1	**1.66**	1	1.64	1	1.65
	10	10	1	1.25	1	**1.54**	1	1.49	1	1.41
	20	12	1	1.03	1	**0.82**	1	**0.80**	1	**1.05**
f4	2	6	1	1.28	1	1.64	1	1.63	1	**1.65**
	5	8	1	1.30	1	**1.72**	1	1.70	1	1.69
	10	10	1	1.24	1	1.36	1	**1.37**	1	1.29
	20	12	1	1.09	0.6	**0.49**	0.6	**0.50**	1	**1.10**
f5	2	6	1	1.28	1	**1.70**	1	1.68	1	**1.70**
	5	8	1	1.26	1	**2.01**	1	1.93	1	1.93
	10	10	1	1.27	1	**2.24**	1	2.10	1	2.08
	20	12	0.84	1.34	1	**3.28**	1	3.08	1	2.89
f6	2	6	1	1.35	1	**1.74**	1	1.69	1	1.67
	5	8	1	1.41	1	**1.82**	1	1.75	1	1.66
	10	10	1	1.37	1	1.76	1	**1.78**	1	1.54
f7	2	6	1	1.48	1	**2.15**	1	2.13	1	2.07
	5	8	0.96	1.47	0.96	**2.24**	1	2.23	0.96	2.02
	10	10	0.96	1.63	0.96	**2.06**	0.92	1.86	0.96	1.81
f8	2	6	1	1.29	1	**1.66**	1	1.64	0.96	1.55
	5	8	1	1.44	1	**1.96**	1	1.91	1	1.91
	10	10	1	1.34	1	**1.79**	1	1.74	1	1.59
f9	2	12	1	1.29	1	1.42	1	**1.43**	1	1.41
	5	16	1	1.31	1	**1.52**	1	1.51	1	1.51
	10	20	1	1.25	1	**1.49**	0.96	1.40	1	1.39
f10	2	12	0.96	1.17	0.52	**0.71**	0.4	**0.56**	0.92	**1.22**
	5	16	1	1.33	1	**1.65**	1	1.61	1	1.47
	10	20	1	1.40	1	**1.75**	1	1.70	1	1.46
f11	2	12	1	1.30	1	**1.50**	1	1.48	1	1.44
	5	16	1	1.31	0.96	1.50	1	**1.56**	0.92	1.41
	10	20	0.92	1.51	0.92	**1.81**	0.88	1.69	0.8	1.50
f12	2	50	0.92	1.59	0.8	1.40	0.88	1.55	0.96	**1.67**
	5	140	0.84	1.34	0.64	1.10	0.8	1.36	0.84	**1.39**

Table 4.8 Average success rate and speedup performance of KA-CMA-ES using pre-selection without and with model impact control, including PS-Interval, CPS-Q_w, CPS-$Q_{\text{selection}}$ and CPS-ρ_{rank}.

Category	Average Success Rate (SR)			
	PS-Interval	CPS-Q_w	CPS-$Q_{\text{selection}}$	CPS-ρ_{rank}
D=2	0.990	0.943	0.940	0.987
D=5	0.983	0.963	0.983	0.977
D=10	0.989	0.989	0.978	0.978
D=20	0.960	0.920	0.920	1.000
Unimodal	0.989	0.982	0.982	0.997
Multimodal	0.974	0.917	0.923	0.957
Overall	0.984	0.959	0.961	0.983
Category	Average Speedup (SPU)			
	PS-Interval	CPS-Q_w	CPS-$Q_{\text{selection}}$	CPS-ρ_{rank}
D=2	1.318	1.554	1.540	1.594
D=5	1.321	1.687	1.695	1.637
D=10	1.335	1.724	1.662	1.561
D=20	1.145	1.549	1.490	1.581
Unimodal	1.277	1.706	1.669	1.651
Multimodal	1.347	1.518	1.510	1.493
Overall	1.302	1.640	1.614	1.596

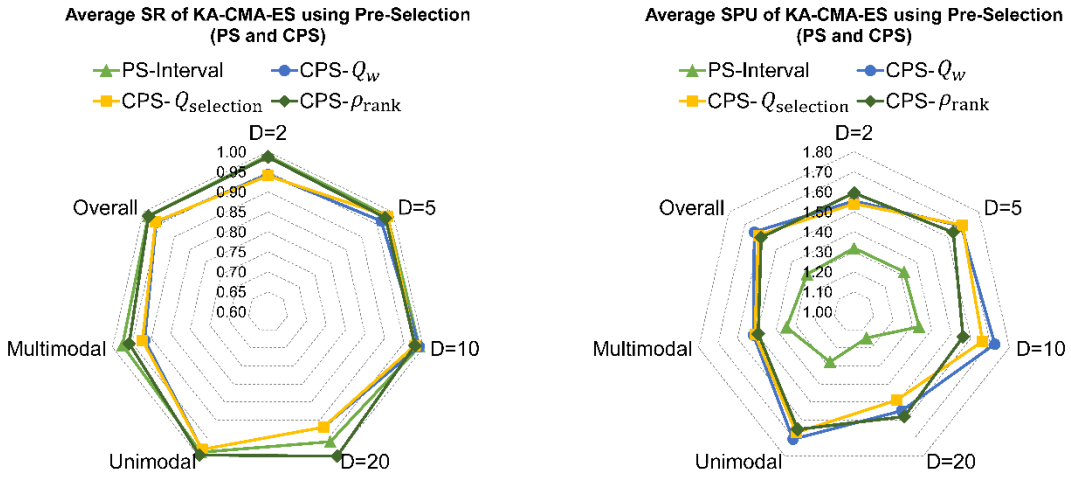

Figure 4.5 Average success rate and speedup of pre-selection without and with model impact control.

4.5.3 Experiments on KA-CMA-ES using Individual-based Control

4.5.3.1 Fixed Individual-based Control (FIC) using Different Metrics

The results of KA-CMA-ES using Fixed Individual-based Control (FIC) with different metrics (Mean, SD, SLB, POI and EI) are given in Table 4.9. In all the tables of experiments results in this chapter, we mark the SPU that do not exceed one ($SPU \leq 1$) in red bold format, which means the corresponding algorithm does not improve the success performance on the corresponding test problem, and mark the maximum SPU of each row in bold number, which represents the best success performance among the investigated algorithms on the corresponding problem. Additionally, the average success rate and speedup performance of FIC using five different metrics on each category problems and overall problems are presented in Table 4.10 and plotted in Figure 4.6. In the tables, FIC-Mean stands for fixed individual-based control using Mean as metric, FIC-SD denotes FIC using SD as metric, FIC-SLB means using SLB as metric in fixed individual-based control, FIC-POI signifies the FIC using POI metric, and FIC-EI represents fixed individual-based control using EI as metric.

It is no doubt that the FIC-EI, on the whole, has the highest success rate among the algorithms of FIC using five metrics. With regards to the success performance and speedup performance, FIC-EI also has the best performance. Among the five metric, these metric (SLB, POI and EI) which balances the exploitation of the surrogate and exploration of search space performs better than metrics that only consider exploitation or exploration (Mean, and SD). Taking both SR and SP into consideration, FIC-EI has advantages over other metrics and thus is suggested to be used in other KA-CMA-ES algorithms using metric, such as KA-CMA-ES using approximate ranking procedure which will be investigated in section 4.5.4.

Table 4.9 Results of KA-CMA-ES using Fixed Individual-based Control (FIC) with different metrics.

Function	D	λ	FIC-Mean			FIC-SD			FIC-SLB			FIC-POI			FIC-EI			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	171	**1.94**	1	182	1.82	1	180	1.84	1	177	1.87	1	174	1.90	1	331
	5	8	1	487	1.82	1	546	1.63	1	481	1.85	1	490	1.81	1	475	**1.87**	1	888
	10	10	1	1102	1.62	1	1429	1.25	1	1085	**1.64**	1	1100	1.62	1	1087	1.64	1	1784
	20	12	1	2462	**1.34**	1	3295	**1.00**	1	2499	1.32	1	2512	1.31	1	2492	1.32	1	3292
f2	2	6	1	203	1.91	1	209	1.85	1	204	1.90	1	202	**1.92**	1	204	1.90	1	387
	5	8	1	578	1.95	1	669	1.69	1	648	1.74	1	569	**1.98**	1	620	1.82	1	1128
	10	10	1	1440	**2.04**	1	1578	1.86	1	1659	1.77	1	1557	1.89	1	1543	1.91	1	2941
	20	12	1	3909	1.91	1	4129	1.81	1	3824	1.95	1	3793	1.97	1	3668	**2.04**	1	7470
f3	2	6	1	188	1.89	1	195	1.83	1	186	**1.91**	1	192	1.85	1	187	1.90	1	356
	5	8	1	535	1.83	1	605	1.62	1	530	**1.85**	1	544	1.80	1	544	1.80	1	980
	10	10	1	1338	1.55	1	1737	1.19	1	1274	**1.62**	1	1289	1.61	1	1281	1.62	1	2070
	20	12	1	3797	1.14	1	4178	1.04	1	3553	1.22	1	3490	1.24	1	3466	**1.25**	1	4327
f4	2	6	1	222	1.88	1	227	1.84	1	226	1.85	1	225	1.85	1	215	**1.94**	1	417
	5	8	1	660	1.85	1	743	1.65	1	653	1.87	1	652	**1.88**	1	661	1.85	1	1224
	10	10	1	1892	1.51	1	2238	1.28	1	1804	1.58	1	1772	**1.61**	1	1810	1.58	1	2858
	20	12	1	6251	1.19	1	6887	1.08	1	5829	1.27	1	5733	**1.30**	1	5788	1.28	1	7431
f5	2	6	1	157	1.76	1	157	1.76	1	155	1.79	1	152	**1.82**	1	154	1.80	1	277
	5	8	1	634	1.81	1	672	1.71	1	625	1.84	1	614	**1.87**	1	649	1.77	1	1149
	10	10	1	1891	**1.88**	0.92	2301	1.55	0.96	1977	1.80	0.96	2030	1.75	0.96	1957	1.82	1	3556
	20	12	0.72	7843	1.79	0.8	8390	1.67	0.6	9279	1.51	0.76	7520	1.86	0.84	7038	**1.99**	0.76	14014
f6	2	6	1	407	1.91	1	410	1.90	1	398	**1.95**	1	422	1.84	1	399	1.95	1	778
	5	8	0.96	1817	1.24	1	1734	1.30	1	1347	1.67	1	1276	**1.77**	1	1341	1.68	1	2253
	10	10	0.16	37375	**0.14**	0.92	7477	**0.68**	0.92	5287	**0.96**	0.8	6229	**0.82**	1	4503	**1.13**	1	5078
f7	2	6	1	360	1.84	1	335	**1.98**	1	348	1.91	1	365	1.82	1	342	1.94	1	663
	5	8	0.96	1284	**1.92**	0.96	1402	1.76	0.92	1343	1.84	0.96	1289	1.91	0.92	1372	1.80	0.96	2468
	10	10	0.84	5455	1.51	0.92	4925	1.67	0.88	4639	1.78	0.76	5055	1.63	0.96	4240	**1.95**	0.8	8248
f8	2	6	1	383	1.87	1	382	1.88	1	374	1.92	0.92	401	1.79	1	373	**1.92**	1	717
	5	8	0.96	1023	1.99	0.96	1193	1.71	0.96	1038	1.97	1	977	**2.09**	1	991	2.06	0.92	2040
	10	10	0.84	2630	1.43	0.92	3273	1.15	0.84	2583	1.45	0.76	2875	1.31	1	2221	**1.69**	0.96	3753
f9	2	12	1	250	1.94	1	263	1.85	1	247	**1.97**	1	261	1.86	1	256	1.90	1	486
	5	16	1	745	2.00	1	812	1.83	1	736	**2.02**	0.96	788	1.89	1	747	1.99	0.96	1489
	10	20	0.92	1775	1.72	0.88	2408	1.27	0.88	1876	1.63	0.92	1732	**1.76**	0.88	1846	1.65	1	3049
f10	2	12	0.96	612	1.72	1	580	1.82	1	557	1.89	1	549	**1.92**	1	560	1.88	1	1054
	5	16	0.96	1653	1.90	1	1845	1.70	1	1548	**2.03**	1	1611	1.95	1	1595	1.97	0.96	3137
	10	20	0.64	5384	1.28	0.92	5427	1.27	0.68	5081	1.36	0.88	3970	1.74	0.92	3880	**1.78**	0.92	6893
f11	2	12	0.96	320	1.87	1	312	1.91	0.96	314	1.90	1	301	1.98	1	294	**2.03**	1	597
	5	16	0.84	999	1.70	0.76	1194	1.42	1	824	**2.06**	0.88	967	1.76	0.8	1046	1.63	0.96	1700
	10	20	0.56	3310	1.35	0.8	3098	1.44	0.84	2153	2.07	0.56	3263	1.37	0.96	1880	**2.37**	0.76	4457
f12	2	50	0.8	1361	1.74	0.96	1240	1.91	0.92	1186	**2.00**	0.84	1345	1.76	0.92	1203	1.97	0.68	2372
	5	140	0.4	9520	1.05	0.56	9777	1.02	0.4	10675	**0.94**	0.72	5947	**1.68**	0.72	6066	1.65	0.8	10020

Table 4.10 Average success rate and speedup performance of KA-CMA-ES using fixed individual-based control (FIC) with different metrics.

Category	Average Success Rate (SR)						Average Speed (SPU)				
	FIC-Mean	FIC-SD	FIC-SLB	FIC-POI	FIC-EI	CMA-ES	FIC-Mean	FIC-SD	FIC-SLB	FIC-POI	FIC-EI
D=2	0.977	0.997	0.990	0.980	0.993	0.973	1.857	1.862	1.902	1.858	1.920
D=5	0.923	0.937	0.940	0.960	0.953	0.963	1.757	1.587	1.806	1.866	1.824
D=10	0.815	0.935	0.909	0.876	0.971	0.949	1.456	1.327	1.606	1.554	1.738
D=20	0.944	0.960	0.920	0.952	0.968	0.952	1.473	1.319	1.455	1.536	1.576
Unimodal	0.948	0.982	0.972	0.971	0.988	0.982	1.661	1.554	1.701	1.716	1.747
Multimodal	0.846	0.911	0.891	0.889	0.943	0.923	1.683	1.584	1.800	1.775	1.892
Overall	0.912	0.957	0.944	0.942	0.972	0.961	1.669	1.564	1.736	1.737	1.798

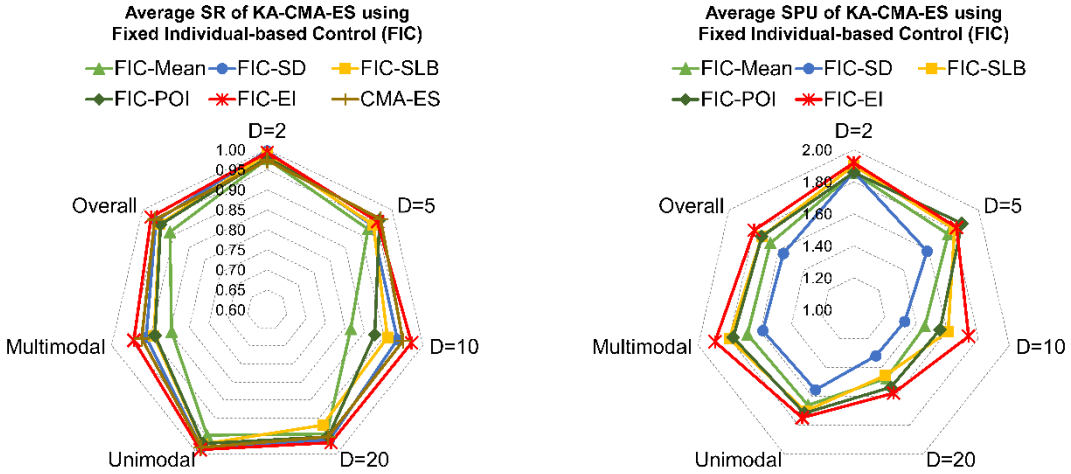

Figure 4.6 Average success rate (SR) and speedup performance (SPU) of KA-CMA-ES using fixed individual-based control with different metrics.

4.5.3.2 Mixed Individual-based Control (MIC)

In the proposed mixed individual-based control (MIC), two metrics, Mean and SLB, can be used. The results of KA-CMA-ES using MIC are given in Table 4.11. The average success rate and speedup performance on each category of problems are listed in Table 4.12 and shown in Figure 4.7. In these tables, MIC-Mean represents mixed individual-based control using Mean as metric, and MIC-SLB means using SLB as metric in mixed individual-based control.

Table 4.11 Results of KA-CMA-ES using Mixed Individual-based Control (MIC).

Function	D	λ	MIC-Mean			MIC-SLB			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	131	**2.53**	1	131	**2.53**	1	331
	5	8	1	395	**2.25**	1	449	1.98	1	888
	10	10	1	1007	**1.77**	1	1140	1.56	1	1784
	20	12	1	2387	**1.38**	1	2590	1.27	1	3292
f2	2	6	1	148	2.61	1	147	**2.63**	1	387
	5	8	1	473	2.38	1	459	**2.46**	1	1128
	10	10	1	1208	**2.43**	1	1256	2.34	1	2941
	20	12	1	3251	**2.30**	1	3664	2.04	1	7470
f3	2	6	1	143	**2.49**	1	143	**2.49**	1	356
	5	8	1	428	**2.29**	1	505	1.94	1	980
	10	10	1	1318	**1.57**	1	1342	1.54	1	2070
	20	12	1	4146	1.04	1	3624	**1.19**	1	4327
f4	2	6	1	165	2.53	1	163	**2.56**	1	417
	5	8	1	535	**2.29**	1	617	1.98	1	1224
	10	10	1	1882	**1.52**	1	1987	1.44	1	2858
	20	12	1	6982	1.06	1	6258	**1.19**	1	7431
f5	2	6	1	114	**2.43**	1	115	2.41	1	277
	5	8	1	500	**2.30**	1	506	2.27	1	1149
	10	10	0.84	1889	**1.88**	0.84	1986	1.79	1	3556
	20	12	0.6	8093	**1.73**	0.6	8140	1.72	0.76	14014
f6	2	6	0.96	323	**2.41**	1	340	2.29	1	778
	5	8	1	1812	1.24	1	1270	**1.77**	1	2253
	10	10	0	∞	**0.0**	0.96	4176	**1.22**	1	5078
f7	2	6	1	265	**2.50**	1	280	2.37	1	663
	5	8	0.92	1138	**2.17**	0.92	1251	1.97	0.96	2468
	10	10	0.84	6086	1.36	0.96	4293	**1.92**	0.8	8248
f8	2	6	0.88	313	2.29	0.96	308	**2.33**	1	717
	5	8	0.92	835	**2.44**	1	842	2.42	0.92	2040
	10	10	0.56	3573	1.05	0.96	2189	**1.71**	0.96	3753
f9	2	12	1	218	**2.23**	1	234	2.08	1	486
	5	16	1	587	**2.54**	0.96	667	2.23	0.96	1489
	10	20	0.88	1590	**1.92**	0.92	1654	1.84	1	3049
f10	2	12	0.64	763	1.38	1	501	**2.10**	1	1054
	5	16	0.84	1464	2.14	0.96	1349	**2.33**	0.96	3137
	10	20	0.2	15116	**0.46**	0.52	5870	**1.17**	0.92	6893
f11	2	12	0.96	270	2.21	1	263	**2.27**	1	597
	5	16	0.84	770	**2.21**	0.8	840	2.02	0.96	1700
	10	20	0.52	3029	1.47	0.64	2584	**1.72**	0.76	4457
f12	2	50	0.84	1181	2.01	0.84	1165	**2.04**	0.68	2372
	5	140	0.24	12172	**0.82**	0.4	8568	**1.17**	0.8	10020

Table 4.12 Average success rate (SR) and speedup performance (SPU) of KA-CMA-ES using mixed individual-based control.

Category	Average Success Rate (SR)			Average Speedup (SPU)	
	MIC-Mean	MIC-SLB	CMA-ES	MIC-Mean	MIC-SLB
D=2	0.940	0.983	0.973	2.302	2.341
D=5	0.897	0.920	0.963	2.090	2.046
D=10	0.713	0.891	0.949	1.449	1.661
D=20	0.920	0.920	0.952	1.503	1.483
Unimodal	0.929	0.972	0.982	1.961	1.957
Multimodal	0.737	0.854	0.923	1.798	1.960
Overall	0.862	0.931	0.961	1.904	1.958

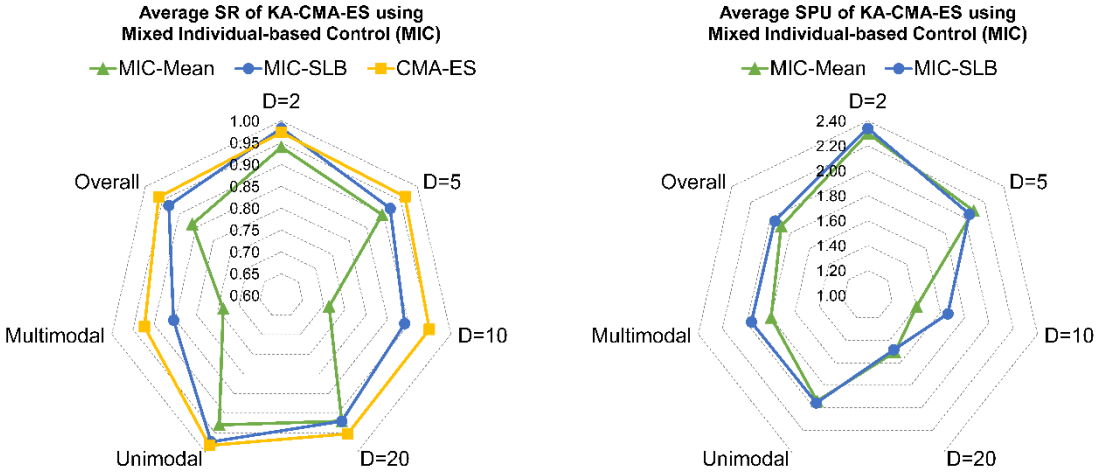

Figure 4.7 Average success rate (SR) and speedup performance (SPU) of KA-CMA-ES using mixed individual-based control (MIC).

In Table 4.11, in terms of speedup performance, MIC-Mean (mixed individual-based control using metric Mean) performs worsen than standard CMA-ES on 10 dimensional f_6 and f_{10} and 2 dimensional f_{12}. While, MIC-SLB (mixed individual-based control using metric SLB) has better success performance than standard CMA-ES on all the test problems. Form this, it can be found that MIC-SLB is more stable and reliable than MIC-Mean.

From Figure 4.7, it is apparent that MIC-SLB has higher success rate (SR) than that of MIC-Mean, especially on multimodal problems. Furthermore, generally, MIC-SLB has better average speedup performance (SPU) than MIC-Mean on all categories of problem except problems with *D*=5. Taking both the success rate and speedup performance into consideration, it can be concluded that MIC-SLB outperforms MIC-Mean.

4.5.3.3 Comparison of Fixed and Mixed Individual-based Control

We compare the performance of KA-CMA-ES using Fixed Individual-based Control (FIC) and KA-CMA-ES using Mixed Individual-based Control (MIC). The success rate (SR) and speedup performance (SPU) of KA-CMA-ES using FIC and MIC are listed and compared in Table 4.13. The average SR and SPU of each category problems of KA-CMA-ES using FIC and MIC are given in Table 4.14 and shown in Figure 4.8.

From Table 4.13, it can be seen that, for most of the problems, KA-CMA-ES using MIC (the proposed mixed individual-based control) has higher speedup performance (better success performance) than using FIC (fixed individual-based control). It worth to note that for 10 dimensional f_6, only FIC-EI and MIC-SLB perform better than the standard CMA-ES, and MIC-SLB also outperforms FIC-EI on this problem. Therefore, the proposed mixed individual-based control (MIC) has better success performance than fixed individual-based control (FIC).

Additionally, from the radar chart of average SR in the left of Figure 4.8, generally, KA-CMA-ES using FIC have higher success rate than that using MIC. The FIC-Mean and MIC-Mean has the lowest success rate among all the investigated algorithms using individual-based control. The FIC-EI generally has the highest success rate. Other algorithms have moderate average success rate larger than 0.85. However, KA-CMA-ES using MIC significantly outperform FIC according to the speedup performance and success performance. Particularly, MIC-SLB has the highest average SPU and acceptable success rate. Therefore, MIC-SLB is preferable among algorithms of KA-CMA-ES using individual-based control.

Table 4.13 Success rate (SR) and speedup performance (SPU) comparison of KA-CMA-ES using Fixed Individual-based Control (FIC) and Mixed Individual-based Control (MIC).

Function	D	λ	FIC-Mean		FIC-SD		FIC-SLB		FIC-POI		FIC-EI		MIC-Mean		MIC-SLB	
			SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU
f1	2	6	1	1.94	1	1.82	1	1.84	1	1.87	1	1.90	1	**2.53**	1	**2.53**
	5	8	1	1.82	1	1.63	1	1.85	1	1.81	1	1.87	1	**2.25**	1	1.98
	10	10	1	1.62	1	1.25	1	1.64	1	1.62	1	1.64	1	**1.77**	1	1.56
	20	12	1	1.34	1	**1.00**	1	1.32	1	1.31	1	1.32	1	**1.38**	1	1.27
f2	2	6	1	1.91	1	1.85	1	1.90	1	1.92	1	1.90	1	2.61	1	**2.63**
	5	8	1	1.95	1	1.69	1	1.74	1	1.98	1	1.82	1	2.38	1	**2.46**
	10	10	1	2.04	1	1.86	1	1.77	1	1.89	1	1.91	1	**2.43**	1	2.34
	20	12	1	1.91	1	1.81	1	1.95	1	1.97	1	2.04	1	**2.30**	1	2.04
f3	2	6	1	1.89	1	1.83	1	1.91	1	1.85	1	1.90	1	**2.49**	1	**2.49**
	5	8	1	1.83	1	1.62	1	1.85	1	1.80	1	1.80	1	**2.29**	1	1.94
	10	10	1	1.55	1	1.19	1	**1.62**	1	1.61	1	1.62	1	1.57	1	1.54
	20	12	1	1.14	1	1.04	1	1.22	1	1.24	1	**1.25**	1	1.04	1	1.19
f4	2	6	1	1.88	1	1.84	1	1.85	1	1.85	1	1.94	1	2.53	1	**2.56**
	5	8	1	1.85	1	1.65	1	1.87	1	1.88	1	1.85	1	**2.29**	1	1.98
	10	10	1	1.51	1	1.28	1	1.58	1	**1.61**	1	1.58	1	1.52	1	1.44
	20	12	1	1.19	1	1.08	1	1.27	1	**1.30**	1	1.28	1	1.06	1	1.19
f5	2	6	1	1.76	1	1.76	1	1.79	1	1.82	1	1.80	1	**2.43**	1	2.41
	5	8	1	1.81	1	1.71	1	1.84	1	1.87	1	1.77	1	**2.30**	1	2.27
	10	10	1	1.88	0.92	1.55	0.96	1.80	0.96	1.75	0.96	1.82	0.84	**1.88**	0.84	1.79
	20	12	0.72	1.79	0.8	1.67	0.6	1.51	0.76	1.86	0.84	**1.99**	0.6	1.73	0.6	1.72
f6	2	6	1	1.91	1	1.90	1	1.95	1	1.84	1	1.95	0.96	**2.41**	1	2.29
	5	8	0.96	1.24	1	1.30	1	1.67	1	1.77	1	1.68	1	1.24	1	**1.77**
	10	10	0.16	**0.14**	0.92	**0.68**	0.92	**0.96**	0.8	**0.82**	1	1.13	0	**0.0**	0.96	**1.22**
f7	2	6	1	1.84	1	1.98	1	1.91	1	1.82	1	1.94	1	**2.50**	1	2.37
	5	8	0.96	1.92	0.96	1.76	0.92	1.84	0.96	1.91	0.92	1.80	0.92	**2.17**	0.92	1.97
	10	10	0.84	1.51	0.92	1.67	0.88	1.78	0.76	1.63	0.96	**1.95**	0.84	1.36	0.96	1.92
f8	2	6	1	1.87	1	1.88	1	1.92	0.92	1.79	1	1.92	0.88	2.29	0.96	**2.33**
	5	8	0.96	1.99	0.96	1.71	0.96	1.97	1	2.09	1	2.06	0.92	**2.44**	1	2.42
	10	10	0.84	1.43	0.92	1.15	0.84	1.45	0.76	1.31	1	1.69	0.56	1.05	0.96	**1.71**
f9	2	12	1	1.94	1	1.85	1	1.97	1	1.86	1	1.90	1	**2.23**	1	2.08
	5	16	1	2.00	1	1.83	1	2.02	0.96	1.89	1	1.99	1	**2.54**	0.96	2.23
	10	20	0.92	1.72	0.88	1.27	0.88	1.63	0.92	1.76	0.88	1.65	0.88	**1.92**	0.92	1.84
f10	2	12	0.96	1.72	1	1.82	1	1.89	1	1.92	1	1.88	0.64	1.38	1	**2.10**
	5	16	0.96	1.90	1	1.70	1	2.03	1	1.95	1	1.97	0.84	2.14	0.96	**2.33**
	10	20	0.64	1.28	0.92	1.27	0.68	1.36	0.88	1.74	0.92	**1.78**	0.2	**0.46**	0.52	1.17
f11	2	12	0.96	1.87	1	1.91	0.96	1.90	1	1.98	1	2.03	0.96	2.21	1	**2.27**
	5	16	0.84	1.70	0.76	1.42	1	2.06	0.88	1.76	0.8	1.63	0.84	**2.21**	0.8	2.02
	10	20	0.56	1.35	0.8	1.44	0.84	2.07	0.56	1.37	0.96	**2.37**	0.52	1.47	0.64	1.72
f12	2	50	0.8	1.74	0.96	1.91	0.92	2.00	0.84	1.76	0.92	1.97	0.84	2.01	0.84	**2.04**
	5	140	0.4	1.05	0.56	1.02	0.4	**0.94**	0.72	**1.68**	0.72	1.65	0.24	**0.82**	0.4	1.17

Table 4.14 Average success rate and speedup performance of KA-CMA-ES using Fixed and Mixed Individual-based Control (FIC and MIC)

Category	Average Success Rate (SR)						
	FIC-Mean	FIC-SD	FIC-SLB	FIC-POI	FIC-EI	MIC-Mean	MIC-SLB
D=2	0.977	0.997	0.990	0.980	0.993	0.940	0.983
D=5	0.923	0.937	0.940	0.960	0.953	0.897	0.920
D=10	0.815	0.935	0.909	0.876	0.971	0.713	0.891
D=20	0.944	0.960	0.920	0.952	0.968	0.920	0.920
Unimodal	0.948	0.982	0.972	0.971	0.988	0.929	0.972
Multimodal	0.846	0.911	0.891	0.889	0.943	0.737	0.854
Overall	0.912	0.957	0.944	0.942	**0.972**	0.862	0.931
Category	Average Speedup Performance (SPU)						
	FIC-Mean	FIC-SD	FIC-SLB	FIC-POI	FIC-EI	MIC-Mean	MIC-SLB
D=2	1.857	1.862	1.902	1.858	1.920	2.302	2.341
D=5	1.757	1.587	1.806	1.866	1.824	2.090	2.046
D=10	1.456	1.327	1.606	1.554	1.738	1.403	1.661
D=20	1.473	1.319	1.455	1.536	1.576	1.503	1.483
Unimodal	1.661	1.554	1.701	1.716	1.747	1.941	1.957
Multimodal	1.683	1.584	1.800	1.775	1.892	1.798	1.960
Overall	1.669	1.564	1.736	1.737	1.798	1.891	**1.958**

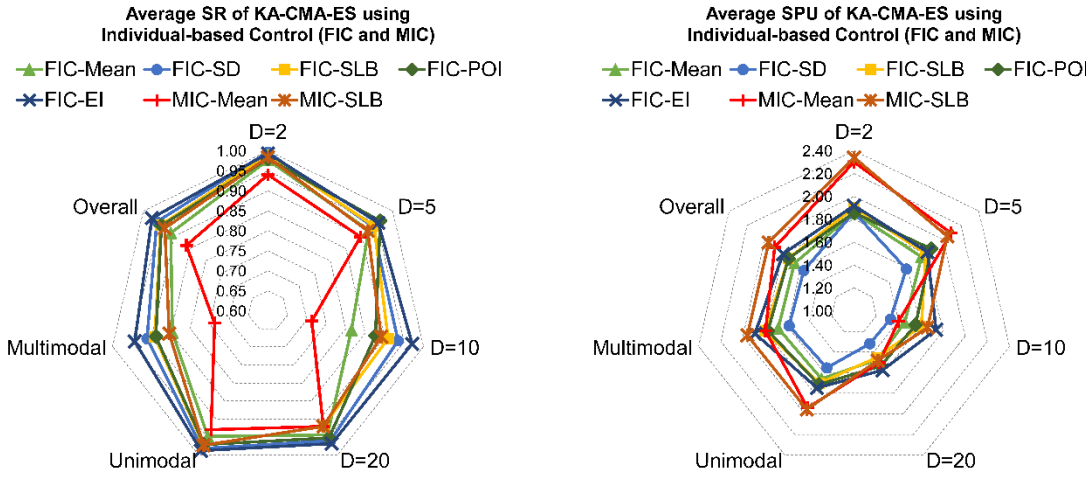

Figure 4.8 Average SR and SPU of KA-CMA-ES using fixed and mixed individual-based control (FIC and MIC).

4.5.4 Experiments on KA-CMA-ES using Approximate Ranking Procedure

The experimental results of KA-CMA-ES using FIC with different metrics have proved that the metric EI is outstanding among the five metrics. Thus, in KA-CMA-ES using

modified approximate ranking procedure (ARP), only two metrics, Mean and EI, are used. The results of KA-CMA-ES using ARP are listed in Table 4.15. The average success rate and speedup performance are given in Table 4.16 and plotted in Figure 4.9. In the tables, ARP-Mean indicates in Mean is used as metric in the ARP, and ARP-EI represents ARP using EI as metric.

From Table 4.15, we can find that ARP-EI (approximate ranking procedure using EI metric) outperforms that using ARP-Mean. On problem of f_6 with D=10, both ARP-Mean and ARP-EI perform worsen than CMA-ES.

According to the the radar chart of average success rate (SR), the average SR of ARP-EI are higher that of ARP-Mean on all categories of problems. In terms of the speedup performance, ARP-EI also performs better than ARP-Mean. Therefore, it can be concluded that ARP-EI outperforms ARP-Mean.

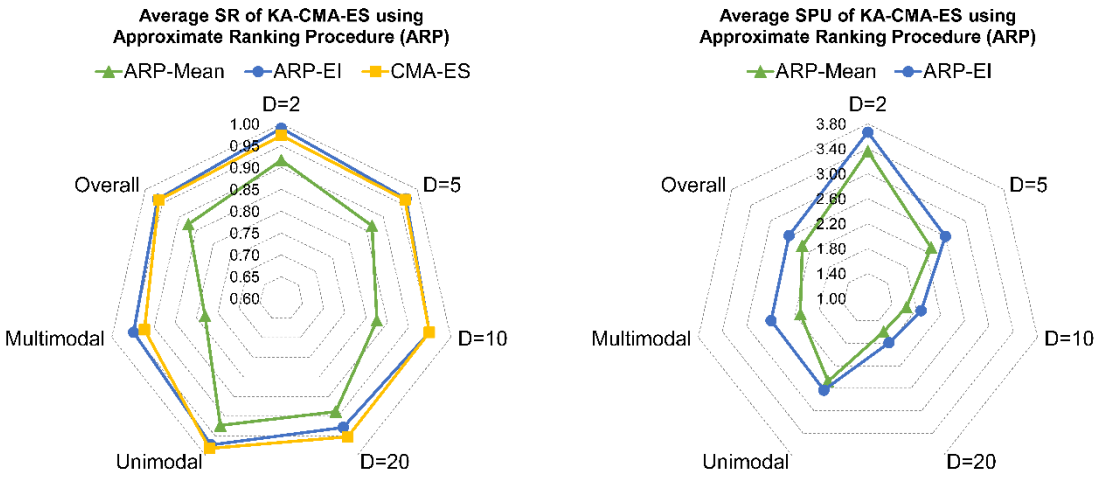

Figure 4.9 Average success rate and speedup performance of KA-CMA-ES using ARP.

Table 4.15 Results of KA-CMA-ES using Approximate Ranking Procedure (ARP)

Function	D	λ	ARP-Mean			ARP-EI			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	82	**4.04**	1	86	3.85	1	331
	5	8	1	351	**2.53**	1	352	2.52	1	888
	10	10	1	1053	1.69	1	1014	**1.76**	1	1784
	20	12	1	2453	1.34	1	2444	**1.35**	1	3292
f2	2	6	1	84	**4.61**	1	87	4.45	1	387
	5	8	1	345	**3.27**	1	365	3.09	1	1128
	10	10	1	1067	2.76	1	996	**2.95**	1	2941
	20	12	1	2530	**2.95**	1	2563	2.91	1	7470
f3	2	6	1	84	**4.24**	1	90	3.96	1	356
	5	8	1	393	2.49	1	387	**2.53**	1	980
	10	10	1	1272	1.63	1	1233	**1.68**	1	2070
	20	12	1	3875	1.12	1	3540	**1.22**	1	4327
f4	2	6	1	100	**4.17**	1	104	4.01	1	417
	5	8	1	490	2.50	1	484	**2.53**	1	1224
	10	10	1	1832	1.56	1	1667	**1.71**	1	2858
	20	12	1	6660	1.12	1	5815	**1.28**	1	7431
f5	2	6	1	76	3.64	1	74	**3.74**	1	277
	5	8	1	424	2.71	1	397	**2.89**	1	1149
	10	10	0.96	1452	**2.45**	0.96	1471	2.42	1	3556
	20	12	0.44	9849	1.42	0.64	6437	**2.18**	0.76	14014
f6	2	6	1	258	3.02	1	193	**4.03**	1	778
	5	8	0.8	1462	1.54	1	1225	**1.84**	1	2253
	10	10	0.12	41047	**0.12**	0.88	6148	**0.83**	1	5078
f7	2	6	1	172	3.85	1	157	**4.22**	1	663
	5	8	0.84	1163	2.12	0.92	968	**2.55**	0.96	2468
	10	10	0.84	5658	1.46	0.88	4249	**1.94**	0.8	8248
f8	2	6	0.84	225	3.19	0.96	191	**3.75**	1	717
	5	8	0.8	906	2.25	1	726	**2.81**	0.92	2040
	10	10	0.88	2400	1.56	0.96	2164	**1.73**	0.96	3753
f9	2	12	1	150	3.24	1	149	**3.26**	1	486
	5	16	0.88	558	2.67	0.96	501	**2.97**	0.96	1489
	10	20	0.92	1717	1.78	1	1585	**1.92**	1	3049
f10	2	12	0.4	980	1.08	1	368	**2.86**	1	1054
	5	16	0.64	1813	1.73	0.96	1146	**2.74**	0.96	3137
	10	20	0.56	6724	1.03	0.96	3903	**1.77**	0.92	6893
f11	2	12	0.88	199	3.00	0.96	182	**3.28**	1	597
	5	16	0.72	752	2.26	0.96	569	**2.99**	0.96	1700
	10	20	0.8	2264	1.97	0.8	2260	**1.97**	0.76	4457
f12	2	50	0.88	1042	2.28	0.96	916	**2.59**	0.68	2372
	5	140	0.72	6363	1.57	0.8	5884	**1.70**	0.8	10020

Table 4.16 Average success rate and speedup of KA-CMA-ES using ARP.

Category	Average Success Rate (SR)			Average Speedup (SPU)	
	ARP-Mean	ARP-EI	CMA-ES	ARP-Mean	ARP-EI
D=2	0.917	0.990	0.973	3.362	3.667
D=5	0.867	0.967	0.963	2.304	2.597
D=10	0.825	0.949	0.949	1.637	1.881
D=20	0.888	0.928	0.952	1.590	1.788
Unimodal	0.923	0.972	0.982	2.475	2.633
Multimodal	0.780	0.949	0.923	2.114	2.597
Overall	0.873	0.964	0.961	2.349	2.620

4.5.5 Experiments on KA-CMA-ES using Generation-based Control

For generation-based control, the Fixed Generation-based Control (FGC) where one generation is controlled in each two generations and the proposed Adaptive Generation-based Control (AGC) are investigated. The results of generation-based control are listed in Table 4.17. The average success rate and speedup performance are presented in Table 4.18 and Figure 4.10. In these tables, FGC stands for fixed generation-based control, AGC-Q_w indicates the adaptive generation-based control based on model quality Q_w, AGC-$Q_{\text{selection}}$ signifies adaptive generation-based control based on model quality measure $Q_{\text{selection}}$, and AGC-ρ_{rank} represents AGC based on model quality measure ρ_{rank}.

In Table 4.17, we mark the SPU that do not exceed one ($\text{SPU} \leq 1$) in red bold format, which means the corresponding algorithm does not improve the success performance on the corresponding test problem. In other words, we can say that the investigated algorithm performs worsen than standard CMA-ES on the test problem if the SPU of the algorithm on a test problem is not larger than 1. For FGC, the number of problems on which SPU is not larger than one is 6. This number for AGC-Q_w is 5, for AGC-$Q_{\text{selection}}$ is 7, and for AGC-ρ_{rank} is 5. The smaller this number is, the corresponding algorithm is more stable and reliable. From this perspective, AGC-Q_w and AGC-ρ_{rank} are slightly more stable and reliable than FGC and AGC-$Q_{\text{selection}}$.

Table 4.17 Results of KA-CMA-ES using FGC and AGC

Function	D	λ	FGC			AGC-Q_w			AGC-$Q_{\text{selection}}$			AGC-ρ_{rank}			CMA-ES	
			SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP	SPU	SR	SP
f1	2	6	1	179	1.85	1	148	**2.24**	1	151	2.19	1	156	2.12	1	331
	5	8	1	523	1.70	1	499	1.78	1	473	**1.88**	1	586	1.52	1	888
	10	10	1	1308	1.36	1	1264	**1.41**	1	1289	1.38	1	1589	1.12	1	1784
	20	12	1	3127	**1.05**	1	3205	1.03	1	3202	1.03	1	3376	**0.98**	1	3292
f2	2	6	1	213	1.82	1	164	**2.36**	1	164	**2.36**	1	170	2.28	1	387
	5	8	1	654	1.72	1	477	2.36	1	460	**2.45**	1	523	2.16	1	1128
	10	10	1	1675	1.76	1	1181	2.49	1	1168	**2.52**	1	1523	1.93	1	2941
	20	12	1	4584	1.63	1	4068	1.84	1	4018	**1.86**	1	4820	1.55	1	7470
f3	2	6	1	195	1.83	1	158	2.25	1	153	**2.33**	1	172	2.07	1	356
	5	8	1	588	1.67	1	549	1.79	1	533	**1.84**	1	650	1.51	1	980
	10	10	1	1635	1.27	1	1630	1.27	1	1616	**1.28**	1	1925	1.08	1	2070
	20	12	1	4958	**0.87**	1	4914	**0.88**	1	4838	**0.89**	1	4523	**0.96**	1	4327
f4	2	6	1	232	1.80	1	179	2.33	1	176	**2.37**	1	195	2.14	1	417
	5	8	1	724	1.69	1	672	1.82	1	658	**1.86**	1	793	1.54	1	1224
	10	10	1	2295	**1.25**	1	2361	1.21	1	2384	1.20	1	2636	1.08	1	2858
	20	12	1	8347	**0.89**	1	8530	**0.87**	1	8030	**0.93**	1	7536	**0.99**	1	7431
f5	2	6	1	155	1.79	1	134	2.07	1	128	**2.16**	1	145	1.91	1	277
	5	8	1	667	1.72	0.96	584	1.97	1	561	**2.05**	1	659	1.74	1	1149
	10	10	0.8	2588	1.37	0.92	1844	**1.93**	0.88	1978	1.80	0.92	2530	1.41	1	3556
	20	12	0.64	10132	1.38	0.72	7362	**1.90**	0.64	8377	1.67	0.8	9835	1.42	0.76	14014
f6	2	6	1	444	1.75	1	349	**2.23**	1	353	2.20	1	432	1.80	1	778
	5	8	1	2098	1.07	0.84	2342	**0.96**	0.84	2182	1.03	1	1802	**1.25**	1	2253
	10	10	0.04	1.19E+05	**0.04**	0	∞	**0.0**	0	∞	**0.0**	1	4881	**1.04**	1	5078
f7	2	6	1	375	1.77	1	296	**2.24**	1	296	**2.24**	1	309	2.15	1	663
	5	8	0.88	1594	**1.55**	0.84	1642	1.50	0.8	1692	1.46	0.88	1762	1.40	0.96	2468
	10	10	0.92	6283	1.31	0.88	6980	1.18	0.96	6248	**1.32**	0.92	6686	1.23	0.8	8248
f8	2	6	1	401	1.79	0.96	329	**2.18**	0.96	337	2.13	1	377	1.90	1	717
	5	8	1	1091	1.87	0.96	1045	**1.95**	0.68	1451	1.41	0.96	1288	1.58	0.92	2040
	10	10	0.64	4173	**0.90**	0.8	3422	1.10	0.64	4314	**0.87**	1	3181	**1.18**	0.96	3753
f9	2	12	1	256	1.90	1	214	**2.27**	1	216	2.25	1	253	1.92	1	486
	5	16	0.88	896	1.66	1	643	**2.32**	0.96	685	2.17	0.96	1058	1.41	0.96	1489
	10	20	0.8	2241	1.36	0.8	2094	**1.46**	0.8	2115	1.44	0.96	2880	1.06	1	3049
f10	2	12	0.92	665	1.58	0.84	631	1.67	0.96	555	**1.90**	0.96	908	1.16	1	1054
	5	16	0.8	2207	**1.42**	0.48	3054	1.03	0.4	3841	**0.82**	1	2714	1.16	0.96	3137
	10	20	0.24	16028	**0.43**	0	∞	**0.0**	0.2	18420	**0.37**	0.88	6927	**1.00**	0.92	6893
f11	2	12	0.96	322	1.85	0.88	275	2.17	0.96	256	**2.33**	1	299	2.00	1	597
	5	16	0.72	1208	1.41	0.76	926	**1.84**	0.72	997	1.71	0.88	1271	1.34	0.96	1700
	10	20	0.64	3146	**1.42**	0.52	3530	1.26	0.4	4690	**0.95**	0.64	4814	**0.93**	0.76	4457
f12	2	50	0.92	1186	2.00	0.8	1184	2.00	0.92	1033	**2.30**	0.84	1219	1.95	0.68	2372
	5	140	0.28	14143	**0.71**	0.12	27611	**0.36**	0.44	7839	**1.28**	0.8	7858	1.28	0.8	10020

Table 4.18 Average success rate and speedup of KA-CMA-ES using FGC and AGC.

Category	Average Success Rate (SR)					Average Speedup (SPU)			
	FGC	AGC-Q_w	AGC-$Q_{\text{selection}}$	AGC-ρ_{rank}	CMA-ES	FGC	AGC-Q_w	AGC-$Q_{\text{selection}}$	AGC-ρ_{rank}
D=2	0.983	0.957	0.983	0.983	0.973	1.810	2.168	2.230	1.949
D=5	0.880	0.830	0.820	0.957	0.963	1.516	1.640	1.662	1.490
D=10	0.735	0.720	0.716	0.938	0.949	1.133	1.210	1.194	1.187
D=20	0.928	0.944	0.928	0.960	0.952	1.166	1.304	1.276	1.179
Unimodal	0.934	0.929	0.928	0.982	0.982	1.458	1.689	1.704	1.553
Multimodal	0.771	0.709	0.717	0.920	0.923	1.450	1.543	1.566	1.418
Overall	0.877	0.852	0.854	0.960	0.961	1.455	1.638	1.656	1.505

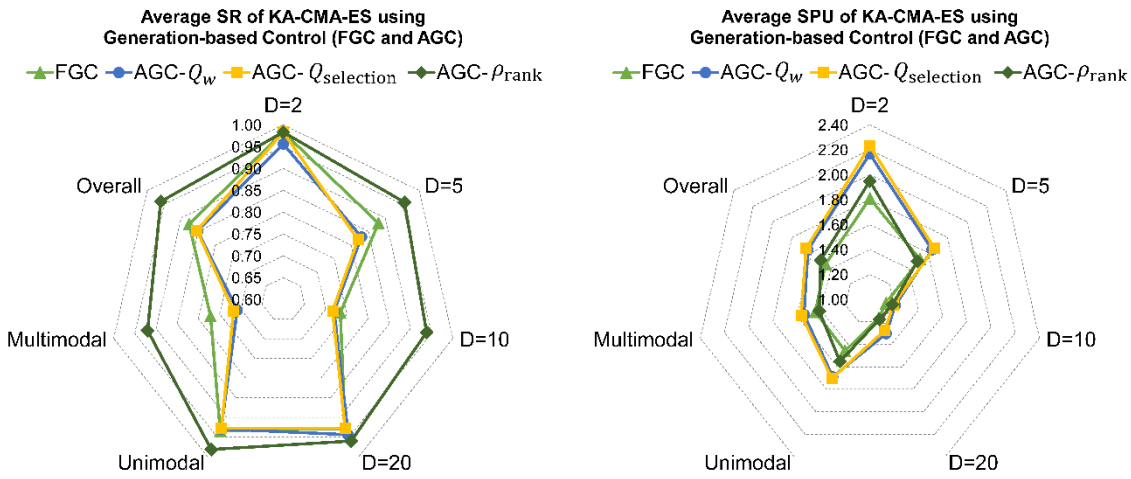

Figure 4.10 Average success rate and speedup of KA-CMA-ES using FGC and AGC.

From the radar charts in Figure 4.10, AGC-ρ_{rank} (adaptive generation control using ρ_{rank}) has the highest average success rate on all the problems. On multimodal problems, the average success rate of FGC, AGC-Q_w and AGC-$Q_{\text{selection}}$ are lower than 0.8. Considering the speedup performance, adaptive generation control using Q_w and $Q_{\text{selection}}$ have better performance. There is slight difference between the average SPU of AGC-Q_w and AGC-$Q_{\text{selection}}$. However, as stated in previous paragraph, AGC-Q_w seems more stable than the AGC-$Q_{\text{selection}}$. Thus, AGC-Q_w is preferable to AGC-$Q_{\text{selection}}$. Overall, AGC-ρ_{rank} is preferable according to success rate and stability, and AGC-Q_w is preferable when speedup performance is considered.

4.5.6 Performance Analysis of All the Investigated KA-CMA-ES Algorithms

In previous three subsections, the KA-CMA-ES using pre-selection, individual-based control, approximate ranking procedure and generation-based control are separately investigated and analyzed. This subsection dedicates to comparing all the investigated algorithms of KA-CMA-ES. Based on previously discussion and results, only the best algorithms of each category of algorithms are chosen as the representatives and used in analysis and comparison here. The PS-Interval is taken as the representative of pre-selection without model impact control (PS). The CPS-Q_w and CPS-ρ_{rank} are considered as typical algorithms of pre-selection with model impact control (CPS). The FIC-EI is used to represent the fixed individual-based control (FIC). The ARP-EI stands for the KA-CMA-ES using approximate ranking procedure (ARP). For generation-based control, AGC-Q_w and AGC-ρ_{rank} are regarded as the representatives.

The success rate (SR) and speedup performance of above selected KA-CMA-ES algorithms are listed in Table 4.19. Additionally, the average success rate (SR) and speedup performance (SPU) of each category of problems are given in Table 4.20 and illustrated by Figure 4.11 and Figure 4.12.

For most test problems, ARP-EI has the best speedup performance. It is apparent that ARP-EI has outstanding speedup performance among all the investigated algorithms. The SPU values of PS-Interval, CPS-Q_w , CPS-ρ_{rank} , FIC-EI and MIC-SLB on all the test problems are larger than one. This shows a stable improvement in success performance of these algorithms. The stabilities of AGC-Q_w and AGC-ρ_{rank} are lower than others. There is only one problem on which ARP-EI performs worsen than CMA-ES. Therefore, it can be stated that the ARP-EI is outstanding among all the investigated KA-CMA-ES algorithms, according to the speedup performance or success performance.

Table 4.19 Performance comparison of KA-CMA-ES algorithms, including PS-Interval, CPS-Q_w, CPS-ρ_{rank}, FIC-EI, MIC-SLB, ARP-EI, AGC-Q_w and AGC-ρ_{rank}.

Function	D	λ	PS-Interval		CPS-Q_w		CPS-ρ_{rank}		FIC-EI		MIC-SLB		ARP-EI		AGC-Q_w		AGC-ρ_{rank}	
			SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU	SR	SPU
f1	2	6	1	1.29	1	1.61	1	1.64	1	1.90	1	2.53	1	**3.85**	1	2.19	1	2.12
	5	8	1	1.26	1	1.64	1	1.63	1	1.87	1	1.98	1	**2.52**	1	1.88	1	1.52
	10	10	1	1.27	1	1.59	1	1.46	1	1.64	1	1.56	1	**1.76**	1	1.38	1	1.12
	20	12	1	1.13	1	1.19	1	1.15	1	1.32	1	1.27	1	**1.35**	1	1.03	1	**0.98**
f2	2	6	1	1.21	1	1.51	1	1.50	1	1.90	1	2.63	1	**4.45**	1	2.36	1	2.28
	5	8	1	1.12	1	1.42	1	1.39	1	1.82	1	2.46	1	**3.09**	1	2.45	1	2.16
	10	10	1	1.17	1	1.58	1	1.65	1	1.91	1	2.34	1	**2.95**	1	2.52	1	1.93
	20	12	0.96	1.13	1	1.98	1	1.73	1	2.04	1	2.04	1	**2.91**	1	1.86	1	1.55
f3	2	6	1	1.30	1	1.62	1	1.60	1	1.90	1	2.49	1	**3.96**	1	2.33	1	2.07
	5	8	1	1.29	1	1.66	1	1.65	1	1.80	1	1.94	1	**2.53**	1	1.84	1	1.51
	10	10	1	1.25	1	1.54	1	1.41	1	1.62	1	1.54	1	**1.68**	1	1.28	1	1.08
	20	12	1	1.03	1	**0.82**	1	1.05	1	**1.25**	1	1.19	1	1.22	1	**0.89**	1	**0.96**
f4	2	6	1	1.28	1	1.64	1	1.65	1	1.94	1	2.56	1	**4.01**	1	2.37	1	2.14
	5	8	1	1.30	1	1.72	1	1.69	1	1.85	1	1.98	1	**2.53**	1	1.86	1	1.54
	10	10	1	1.24	1	1.36	1	1.29	1	1.58	1	1.44	1	**1.71**	1	1.20	1	1.08
	20	12	1	1.09	0.6	**0.49**	1	1.10	1	**1.28**	1	1.19	1	1.28	1	**0.93**	1	**0.99**
f5	2	6	1	1.28	1	1.70	1	1.70	1	1.80	1	2.41	1	**3.74**	1	2.16	1	1.91
	5	8	1	1.26	1	2.01	1	1.93	1	1.77	1	2.27	1	**2.89**	1	2.05	1	1.74
	10	10	1	1.27	1	2.24	1	2.08	0.96	1.82	0.84	1.79	0.96	**2.42**	0.88	1.80	0.92	1.41
	20	12	0.84	1.34	1	**3.28**	1	2.89	0.84	1.99	0.6	1.72	0.64	2.18	0.64	1.67	0.8	1.42
f6	2	6	1	1.35	1	1.74	1	1.67	1	1.95	1	2.29	1	**4.03**	1	2.20	1	1.80
	5	8	1	1.41	1	1.82	1	1.66	1	1.68	1	1.77	1	**1.84**	0.84	1.03	1	1.25
	10	10	1	1.37	1	**1.76**	1	1.54	1	1.13	0.96	1.22	0.88	**0.83**	0	**0.0**	1	1.04
f7	2	6	1	1.48	1	2.15	1	2.07	1	1.94	1	2.37	1	**4.22**	1	2.24	1	2.15
	5	8	0.96	1.47	0.96	2.24	0.96	2.02	0.92	1.80	0.92	1.97	0.92	**2.55**	0.8	1.46	0.88	1.40
	10	10	0.96	1.63	0.96	**2.06**	0.96	1.81	0.96	1.95	0.96	1.92	0.88	1.94	0.96	1.32	0.92	1.23
f8	2	6	1	1.29	1	1.66	0.96	1.55	1	1.92	0.96	2.33	0.96	**3.75**	0.96	2.13	1	1.90
	5	8	1	1.44	1	1.96	1	1.91	1	2.06	1	2.42	1	**2.81**	0.68	1.41	0.96	1.58
	10	10	1	1.34	1	**1.79**	1	1.59	1	1.69	0.96	1.71	0.96	1.73	0.64	**0.87**	1	1.18
f9	2	12	1	1.29	1	1.42	1	1.41	1	1.90	1	2.08	1	**3.26**	1	2.25	1	1.92
	5	16	1	1.31	1	1.52	1	1.51	1	1.99	0.96	2.23	0.96	**2.97**	0.96	2.17	0.96	1.41
	10	20	1	1.25	1	1.49	1	1.39	0.88	1.65	0.92	1.84	1	**1.92**	0.8	1.44	0.96	1.06
f10	2	12	0.96	1.17	0.52	**0.71**	0.92	1.22	1	1.88	1	2.10	1	**2.86**	0.96	1.90	0.96	1.16
	5	16	1	1.33	1	1.65	1	1.47	1	1.97	0.96	2.33	0.96	**2.74**	0.4	**0.82**	1	1.16
	10	20	1	1.40	1	1.75	1	1.46	0.92	**1.78**	0.52	1.17	0.96	1.77	0.2	**0.37**	0.88	**1.00**
f11	2	12	1	1.30	1	1.50	1	1.44	1	2.03	1	2.27	0.96	**3.28**	0.96	2.33	1	2.00
	5	16	1	1.31	0.96	1.50	0.92	1.41	0.8	1.63	0.8	2.02	0.96	**2.99**	0.72	1.71	0.88	1.34
	10	20	0.92	1.51	0.92	1.81	0.8	1.50	0.96	**2.37**	0.64	1.72	0.8	1.97	0.4	**0.95**	0.64	**0.93**
f12	2	50	0.92	1.59	0.8	1.40	0.96	1.67	0.92	1.97	0.84	2.04	0.96	**2.59**	0.92	2.30	0.84	1.95
	5	140	0.84	1.34	0.64	1.10	0.84	1.39	0.72	1.65	0.4	1.17	0.8	**1.70**	0.44	1.28	0.8	1.28

Table 4.20 Comparison of Average success rate (SR) and speedup performance (SPU) of KA-CMA-ES algorithms, including PS-Interval, CPS-Q_w, CPS-ρ_{rank}, FIC-EI, MIC-SLB, ARP-EI, AGC-Q_w and AGC-ρ_{rank}.

Category	Average Success Rate (SR)							
	PS-Interval	CPS-Q_w	CPS-ρ_{rank}	FIC-EI	MIC-SLB	ARP-EI	AGC-Q_w	AGC-ρ_{rank}
D=2	0.990	0.943	0.987	0.993	0.983	0.990	0.957	0.983
D=5	0.983	0.963	0.977	0.953	0.920	0.967	0.830	0.957
D=10	0.989	0.989	0.978	0.971	0.891	0.949	0.720	0.938
D=20	0.960	0.920	1.000	0.968	0.920	0.928	0.944	0.960
Unimodal	0.989	0.982	0.997	0.988	0.972	0.972	0.929	0.982
Multimodal	0.974	0.917	0.957	0.943	0.854	0.949	0.709	0.920
Overall	**0.984**	0.959	0.983	0.972	0.931	0.964	0.852	0.960
Category	Average Speedup (SPU)							
	PS-Interval	CPS-Q_w	CPS-ρ_{rank}	FIC-EI	MIC-SLB	ARP-EI	AGC-Q_w	AGC-ρ_{rank}
D=2	1.318	1.554	1.594	1.920	2.341	3.667	2.168	1.949
D=5	1.321	1.687	1.637	1.824	2.046	2.597	1.640	1.490
D=10	1.335	1.724	1.561	1.738	1.661	1.881	1.210	1.187
D=20	1.145	1.549	1.581	1.576	1.483	1.788	1.304	1.179
Unimodal	1.277	1.706	1.651	1.747	1.957	2.633	1.689	1.553
Multimodal	1.347	1.518	1.493	1.892	1.960	2.597	1.543	1.418
Overall	1.302	1.640	1.596	1.798	1.958	**2.620**	1.638	1.505

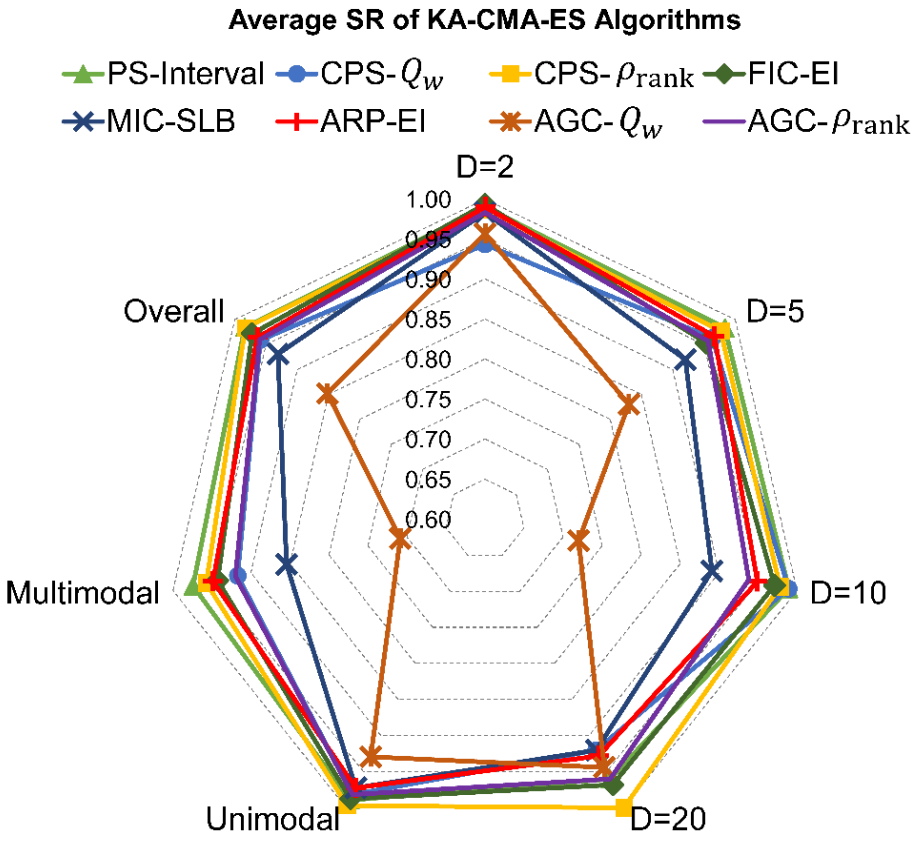

Figure 4.11 Comparison of average success rate (SR) of KA-CMA-ES algorithms.

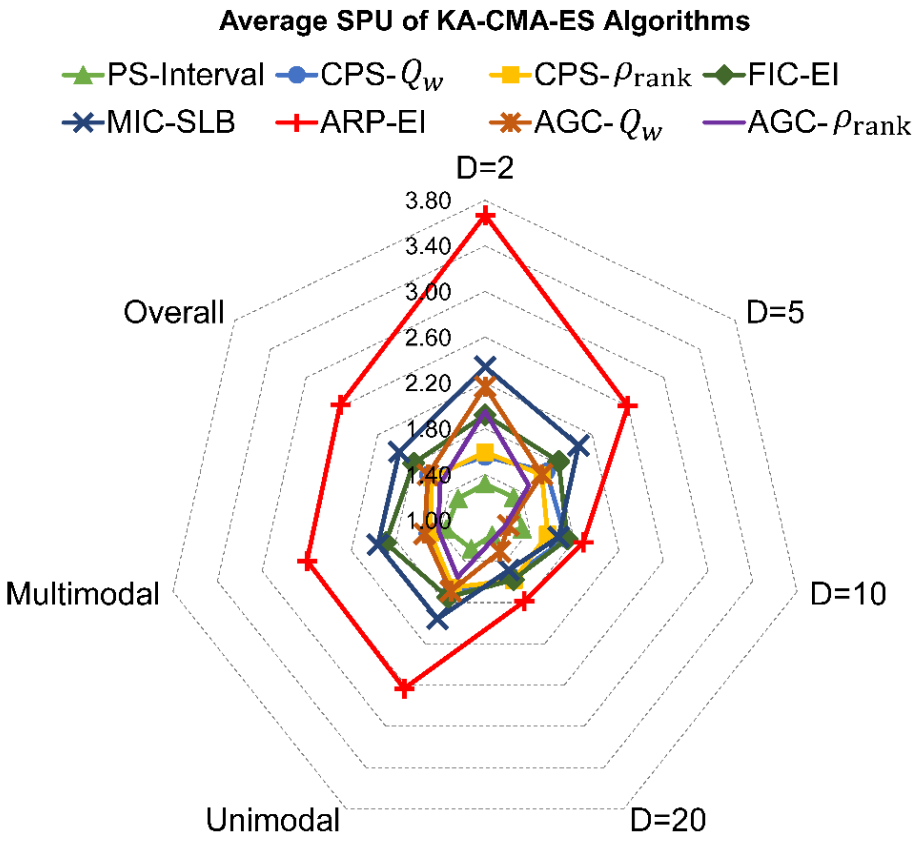

Figure 4.12 Comparison of average speedup performance (SPU) of KA-CMA-ES algorithms.

In Figure 4.11, it can be found that AGC-Q_w generally has the lowest average success rate. The average success rates of PS-Interval, CPS-Q_w, CPS-ρ_{rank}, FIC-EI, ARP-EI and AGC-ρ_{rank} of overall test problems are higher than 0.9. From Figure 4.12, it is no doubt that ARP-EI significantly outperforms other algorithms with respect to average SPU. Considering both the average SR and SPU, it can be concluded that ARP-EI is preferable among all the investigate KA-CMA-ES algorithms in this work. Additionally, the proposed MIC-SLB has the second best performance, and FIC-EI has the third best performance. These three algorithms are suggested in KA-CMA-ES.

In order to illustrate the improvement in efficiency of KA-CMA-ES compared with the standard CMA-ES, the convergence graphs are presented. The graphs show the median performance of the 25 runs on each problems. Since so many algorithms are studied in this work, only the convergence graphs of several representative algorithms are plotted. Here we

given the convergence graphs on each problem of the standard CMA-ES, CPS-ρ_{rank}, MIC-SLB, ARP-EI and AGC-ρ_{rank}. The convergence graphs of 12 functions (40 test problems) are given from Figure 4.13 to Figure 4.24. It can be apparently found that the KA-CMA-ES algorithms converge more quickly than the standard CMA-ES for almost all the test problems. It can be stated that KA-CMA-ES algorithms have better convergence rate and outperform the standard CMA-ES. In other words, KA-CMA-ES algorithms are more efficient than CMA-ES. Therefore, for expensive optimization problems, KA-CMA-ES can significantly reduce the computational cost. Furthermore, it can also be found that, among the KA-CMA-ES algorithms, ARP-EI generally has the best performance.

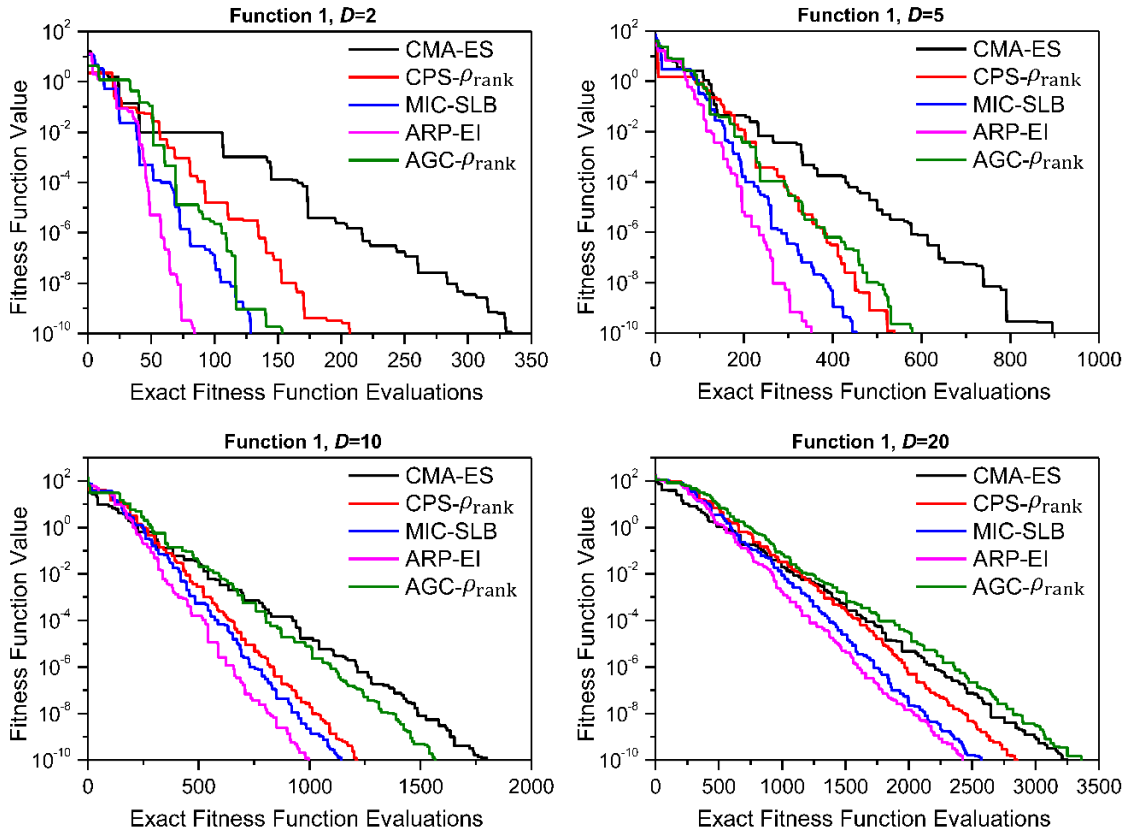

Figure 4.13 Convergence graphs of function 1

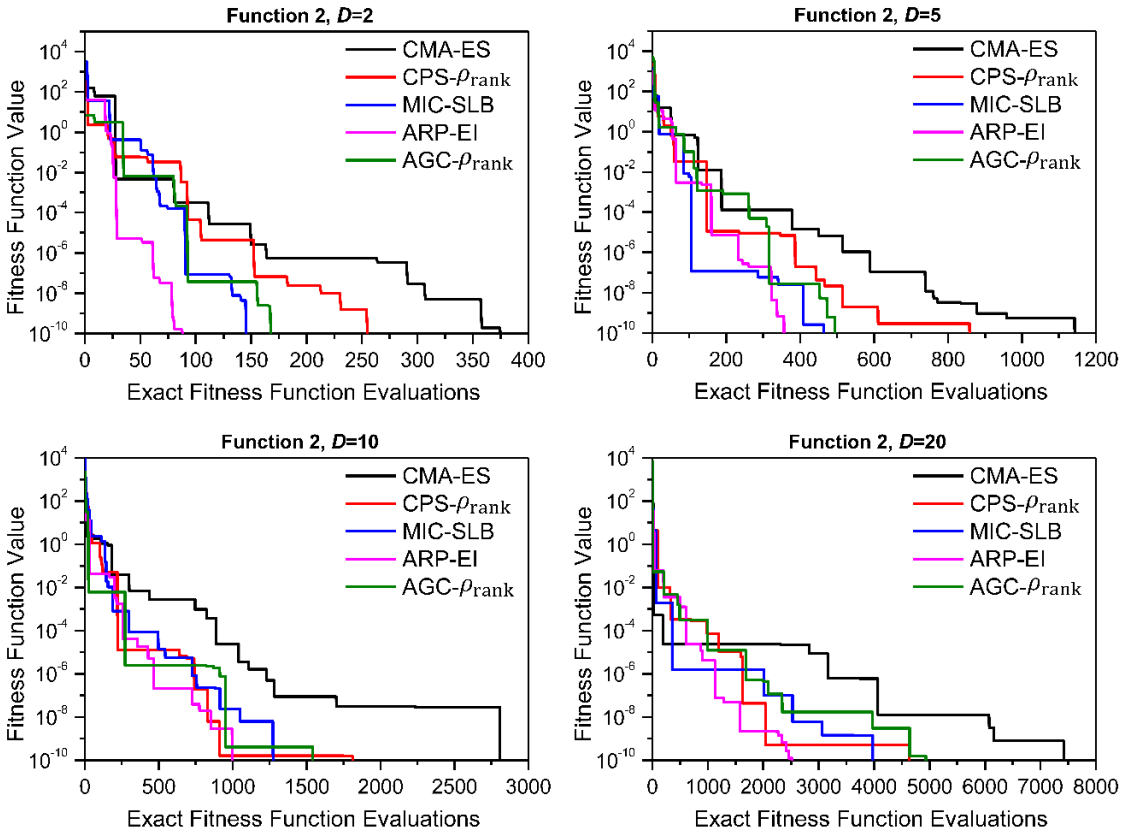

Figure 4.14 Convergence graphs of function 2

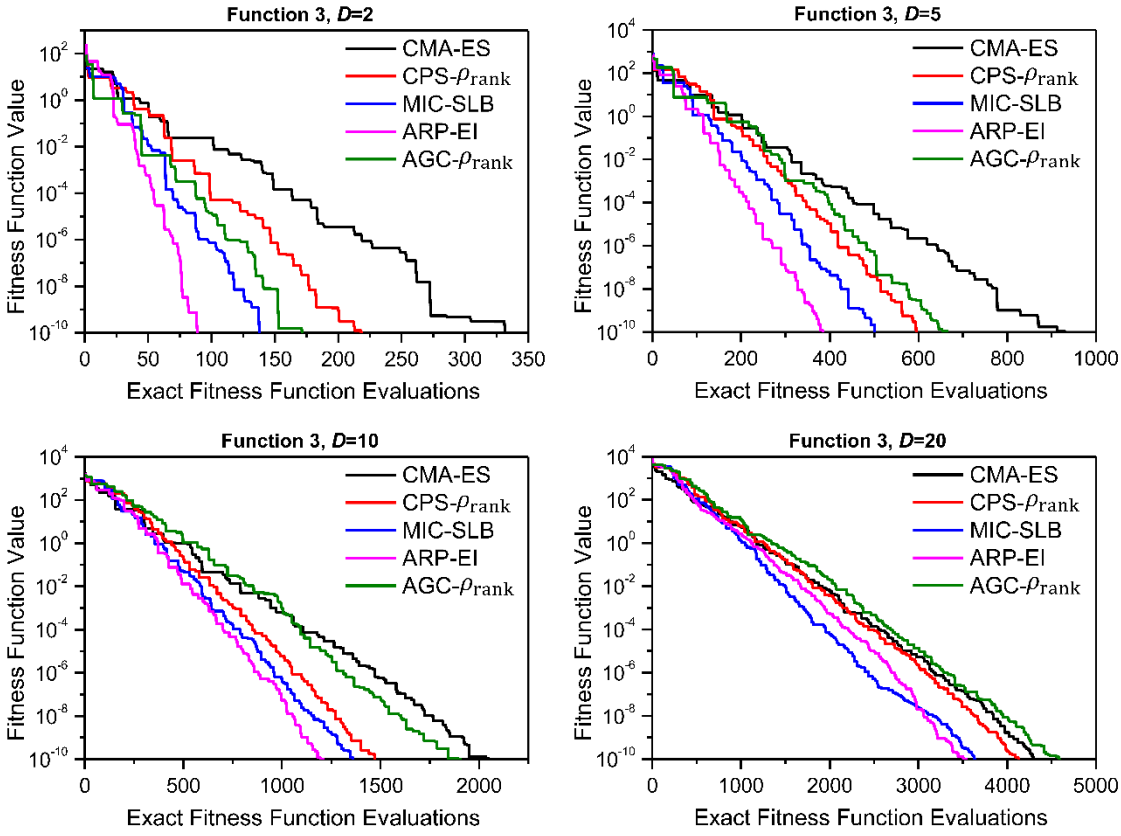

Figure 4.15 Convergence graphs of function 3

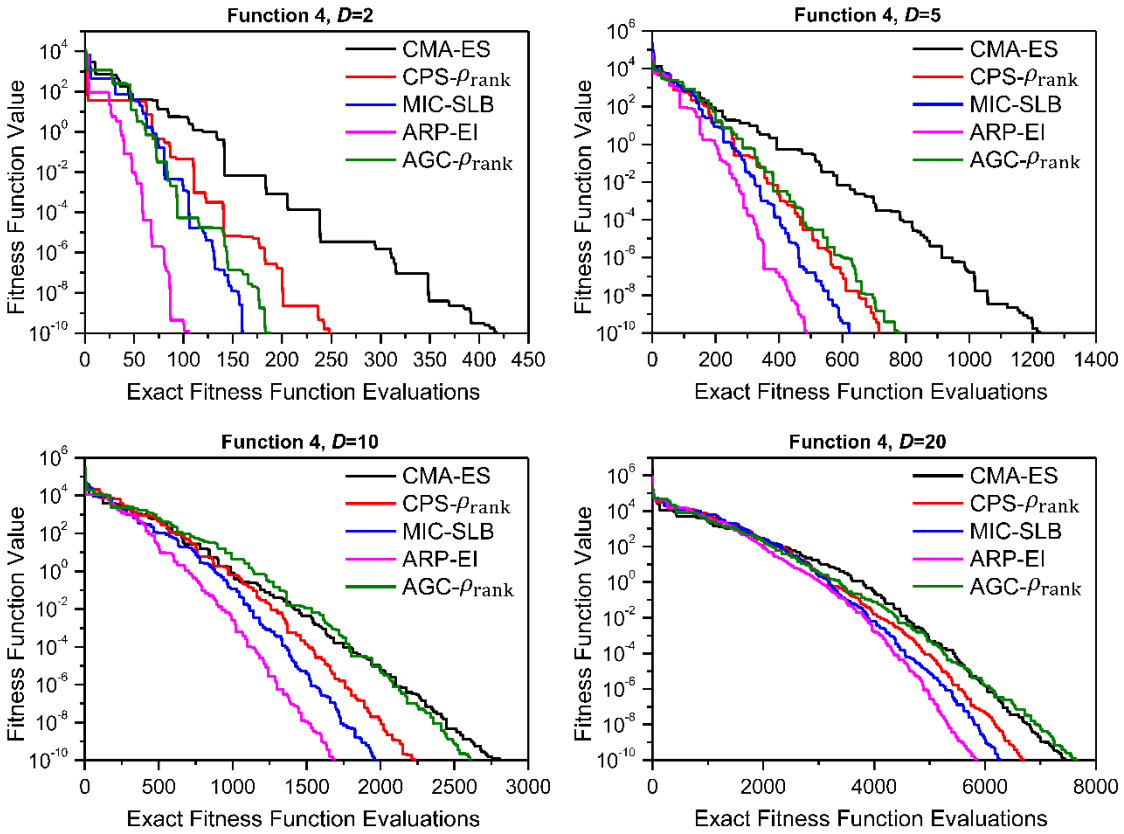

Figure 4.16 Convergence graphs of function 4

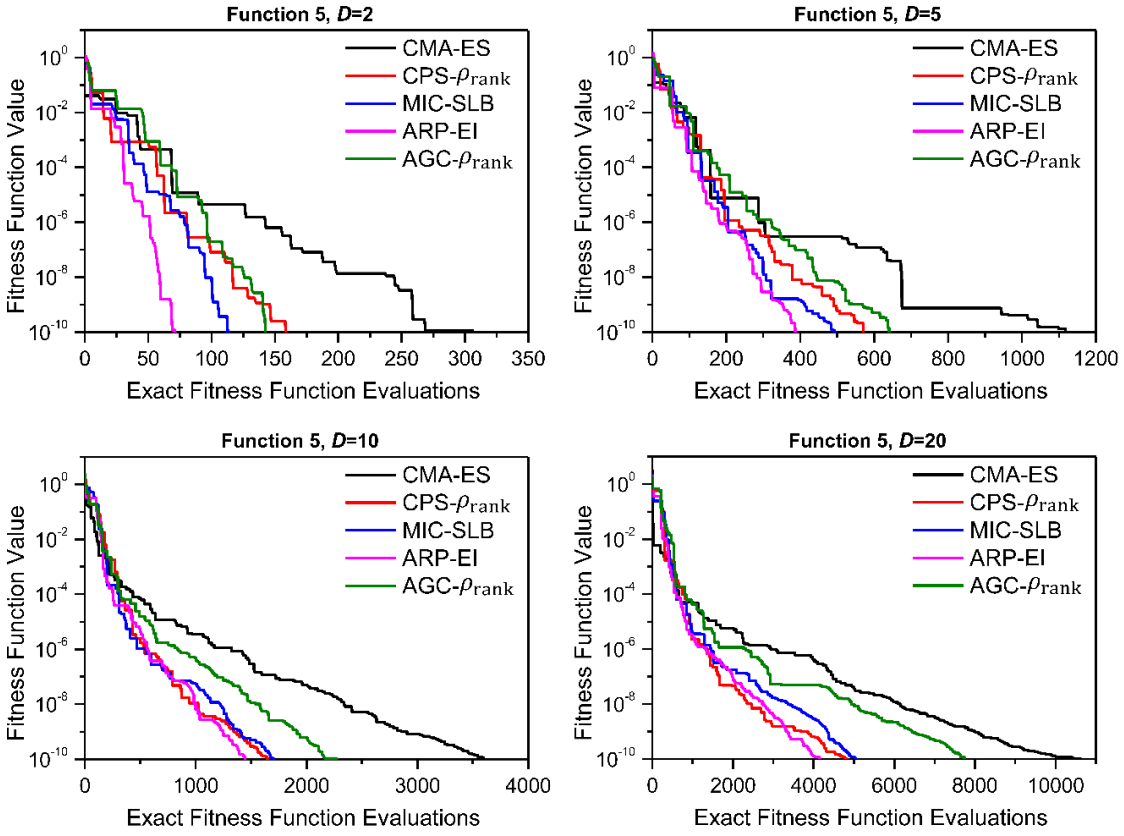

Figure 4.17 Convergence graphs of function 5

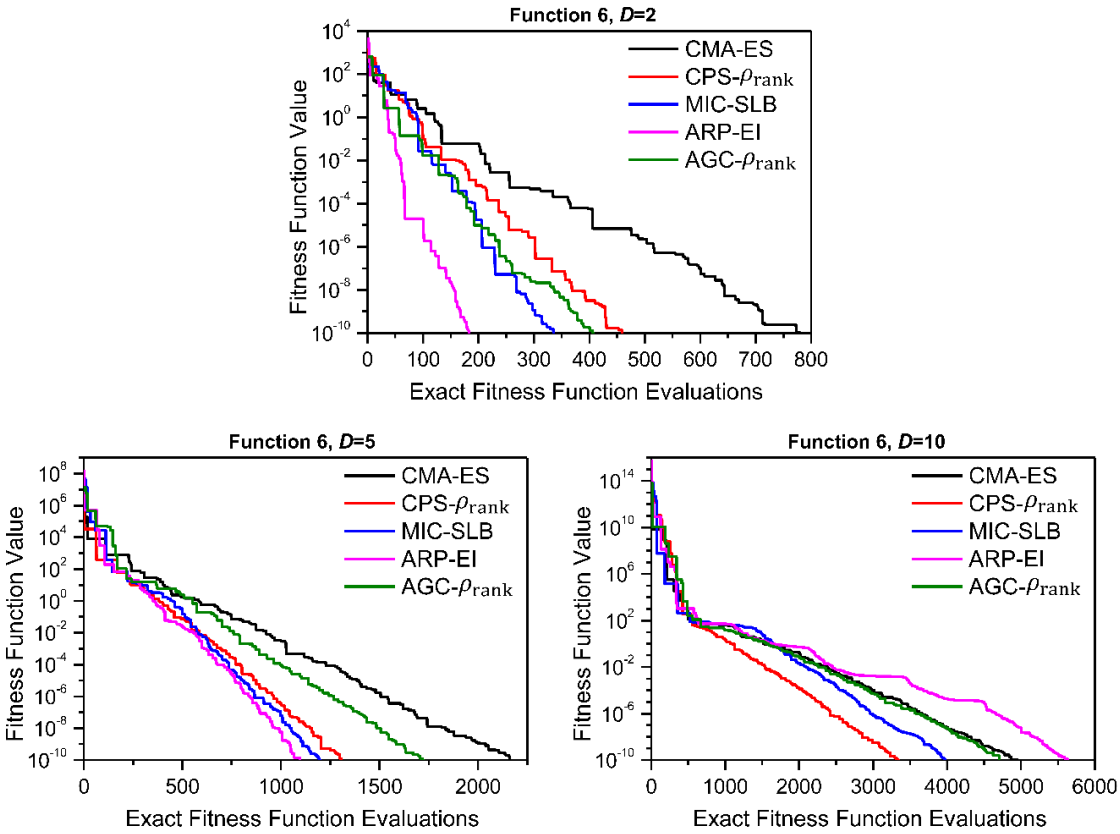

Figure 4.18 Convergence graphs of function 6

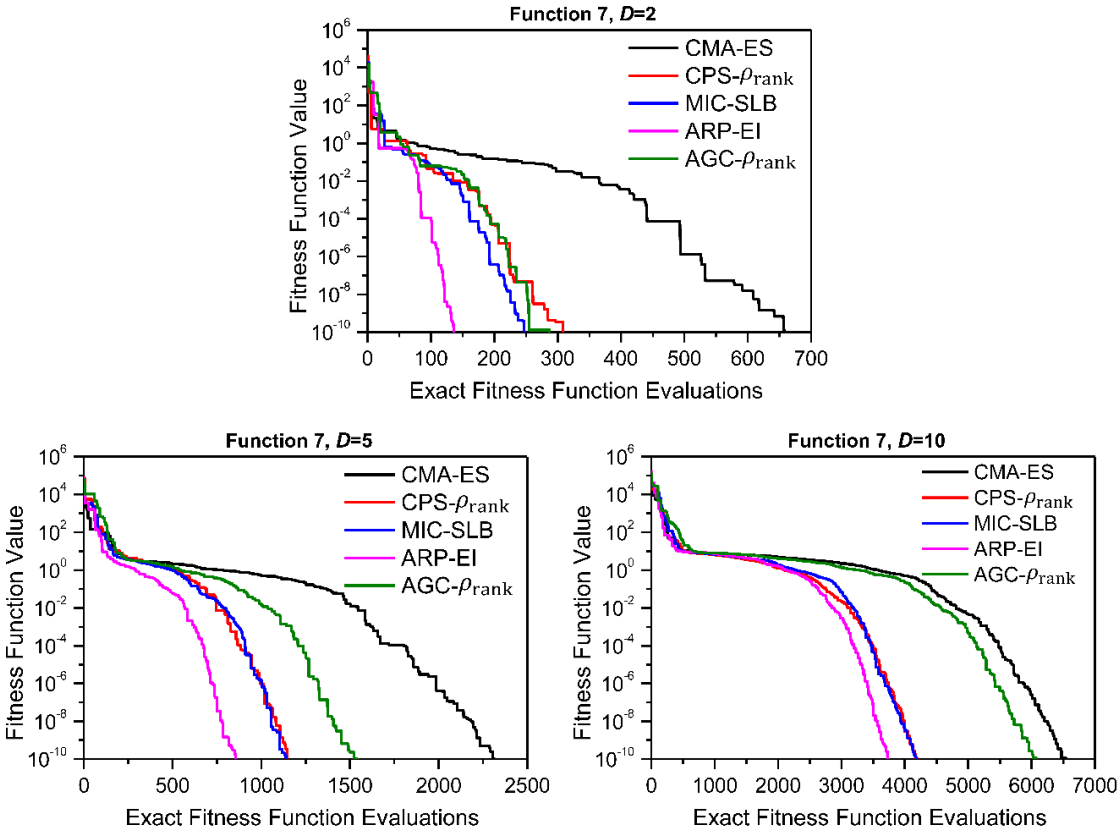

Figure 4.19 Convergence graphs of function 7

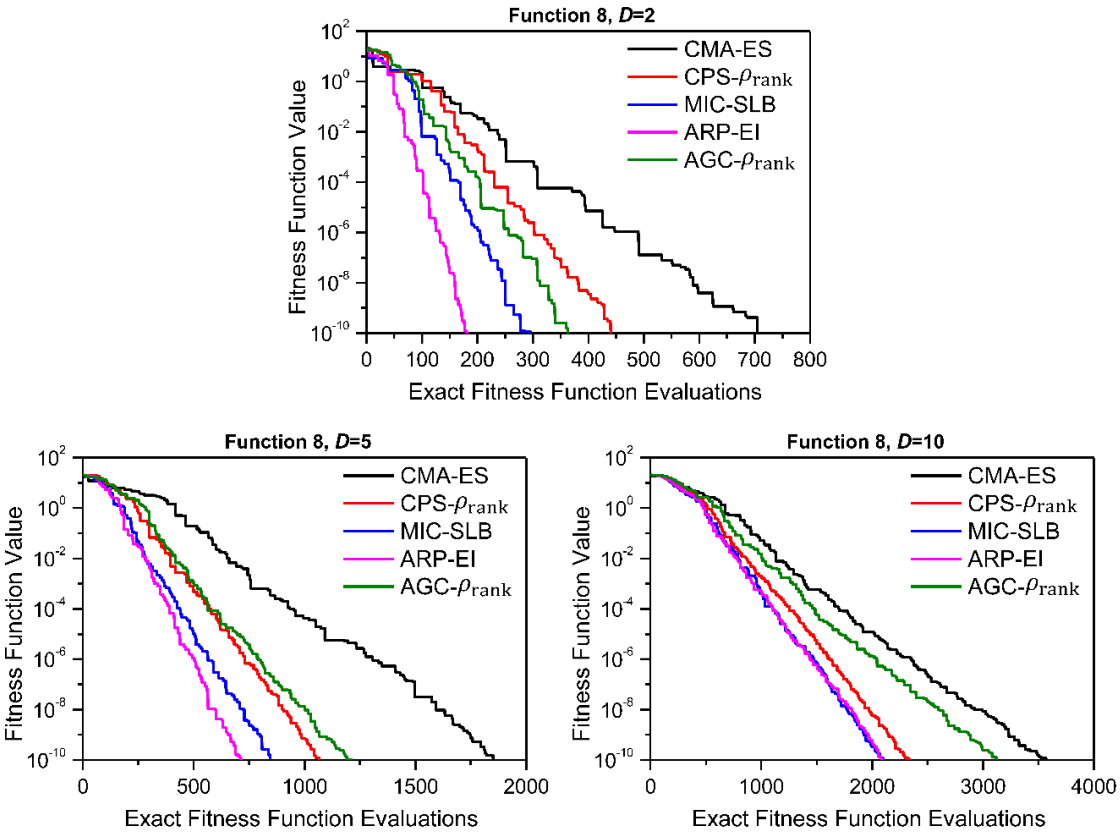

Figure 4.20 Convergence graphs of function 8

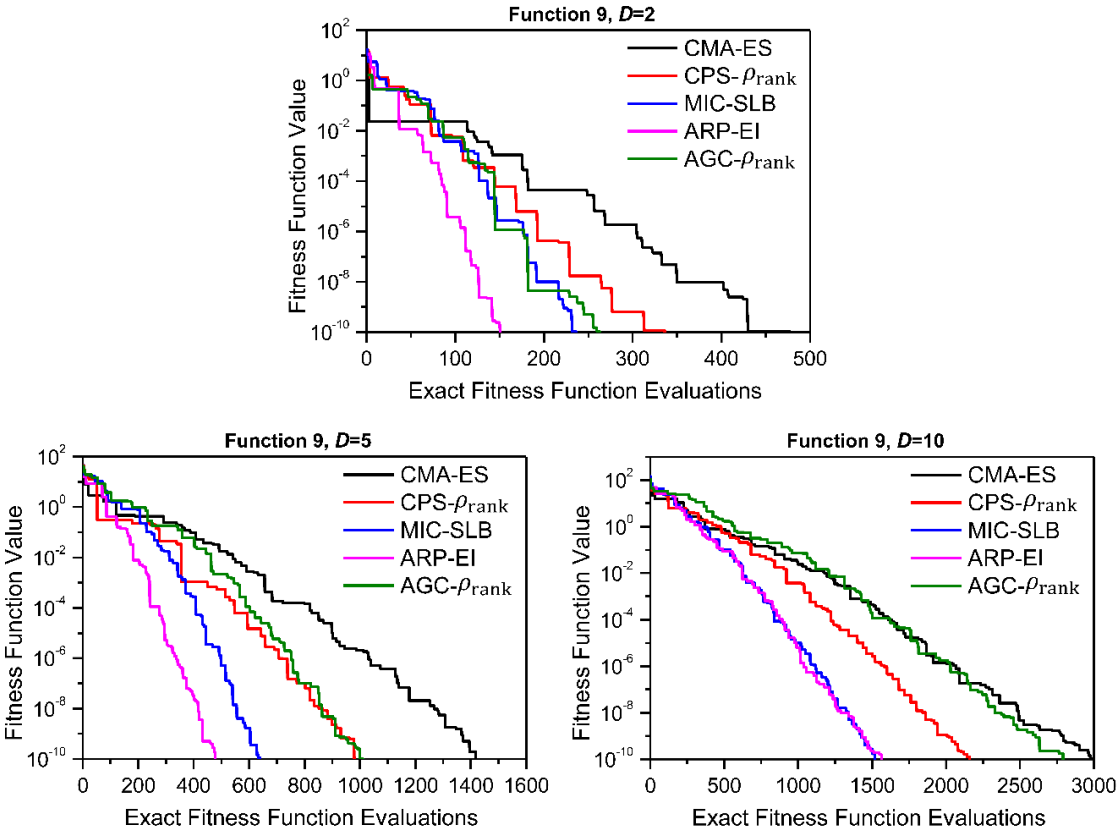

Figure 4.21 Convergence graphs of function 9

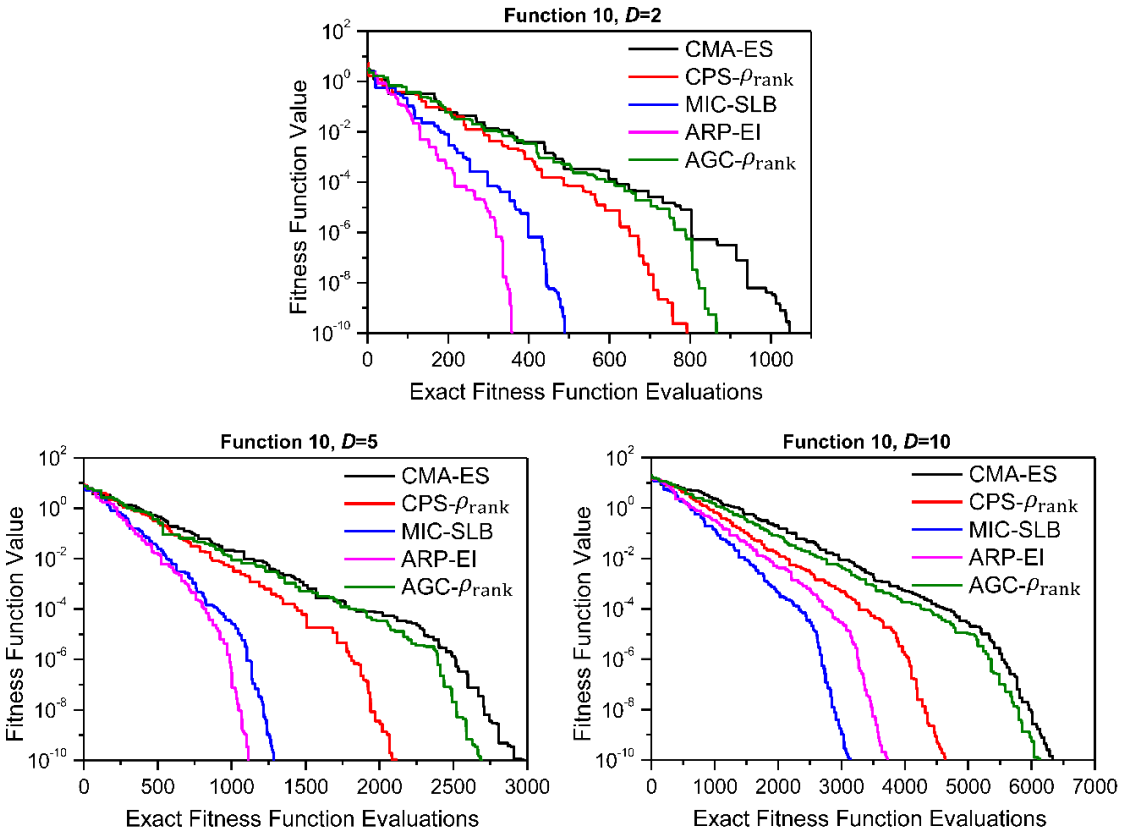

Figure 4.22 Convergence graphs of function 10

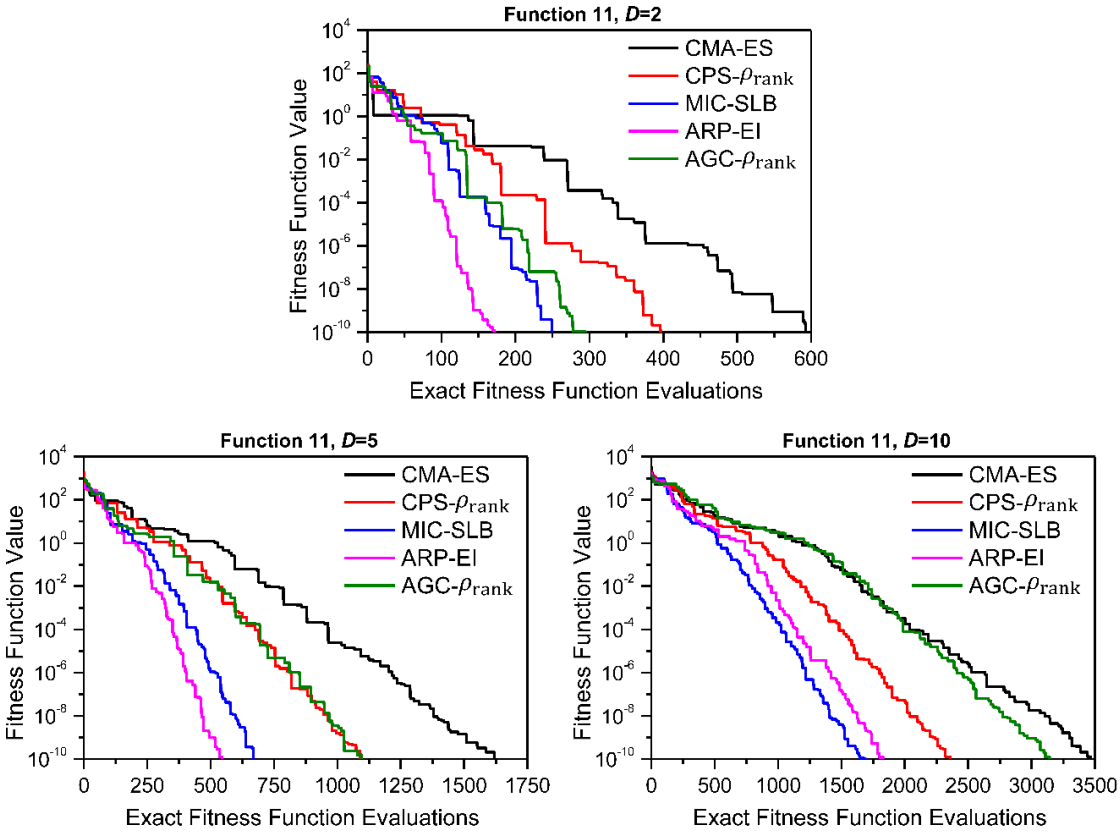

Figure 4.23 Convergence graphs of function 11

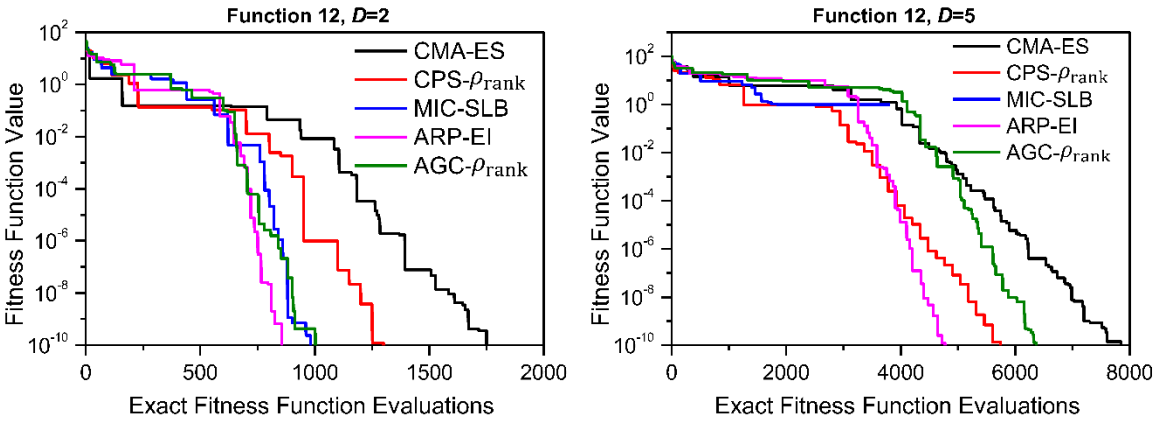

Figure 4.24 Convergence graphs of function 12

4.6 Summary

This chapter focused on Kriging-Assisted CMA-ES (KA-CMA-ES) for expensive optimization problems. New approaches for training set selection and evolution control have been proposed, and the corresponding KA-CMA-ES algorithms are formulated. Experimental studies on 40 test problems are performed to evaluate the performance of the developed KA-CMA-ES algorithms.

The results of experiments on KA-CMA-ES using PS have shown that the proposed confidence interval method for training set selection is superior to the commonly used 'Recently' and 'kNN' methods for training set selection. The experimental results of KA-CMA-ES using CPS have proven that CPS-Q_w has the highest average speedup performance among the investigated algorithms of CPS, while CPS-ρ_{rank} is more stable and has higher success rate. With comparison of KA-CMA-ES using PS and CPS, it is apparent that KA-CMA-ES using CPS significantly perform better than that using PS.

In the experiments of fixed individual-based control (FIC), five different metrics have been investigated. The results demonstrated that the EI metric has the best performance among these five, considering both the success rate and success performance. For the proposed mixed individual-based control (MIC), the MIC using SLB metric outperforms that using Mean as metric. By comparing KA-CMA-ES using FIC and MIC, KA-CMA-ES using MIC significantly outperform FIC according to the speedup performance and success

performance. Particularly, MIC-SLB has the highest average speedup performance and acceptable success rate.

Furthermore, in KA-CMA-ES using the modified approximate ranking procedure (ARP), the Mean and EI metrics have been studied. The results show that ARP-EI clearly outperforms ARP-Mean. This also proves that EI is the most promising metric in evolution control.

The results of experiments of KA-CMA-ES using generation-based control, including fixed generation-based control (FGC) and the proposed adaptive generation-based control (AGC), indicate that the proposed AGC have better performance than FGC. AGC-ρ_{rank} has the highest average success rate on all the problems. Considering the speedup performance, adaptive generation control using Q_w and $Q_{\text{selection}}$ have better performance. Overall, AGC-ρ_{rank} is preferable according to success rate and stability, and AGC-Q_w is preferable when speedup performance is considered.

Considering all the investigated algorithms, it is apparent that ARP-EI has outstanding speedup performance. The SPU values of PS-Interval, CPS-Q_w, CPS-ρ_{rank}, FIC-EI and MIC-SLB on all the test problems are larger than one. This shows a stable improvement in success performance of these algorithms. The stabilities of AGC-Q_w and AGC-ρ_{rank} are lower than others. There is only one problem on which ARP-EI performs worsen than CMA-ES. Therefore, it can be stated that the ARP-EI is outstanding among all the investigated KA-CMA-ES algorithms, according to the speedup performance or success performance.

5. Applications in Material Parameter Identification

This chapter presents applications of the proposed KA-CMA-ES algorithm in material parameter identification, which involves expensive optimization problems. An elastic-plastic damage model, in which material's strain hardening behavior is described by Swift law and ductile damage is modeled by Lemaitre's damage model, is presented and implemented in numerical simulation through VUMAT subroutine in ABAQUS. Then, the KA-CMA-ES using Approximate Ranking Procedure with metric EI (ARP-EI) is applied in inverse method of material parameter identification.

5.1 Introduction

Constitutive model and parameter identification can be seen as the two main aspects of material modeling. Specifically, constitutive model describes material's behavior by the use of mathematical equations or formulations under the framework of constitutive theories; while parameter identification determines the unknown parameters in the constitutive model on the basis of experimental data. Both the constitutive model and parameter identification play important roles in material modeling, and neither of the two aspects could be neglected.

5.1.1 Constitutive Model

The material's constitutive model has experienced huge development during last decades with the improvements both in constitutive theories and their applications. The descriptions of elasticity and plasticity have been comprehensively investigated and theorized in elastic-plastic theories (or elastoplastic theories). Even the more complicated material's behavior, such as viscoplasticity, fatigue, damage and anisotropy, can be modeled at present. With regard to damage, which is a critical and popular issue in engineering

material modeling, currently, continuum damage mechanics (CDM) provides a comprehensive framework for the modeling of material's behavior with consideration of damage from a point view of continuum mechanics [91]. Since an aluminum material is examined in this paper, the ductile damage behavior of the material is modeled by the application of continuum damage mechanic theories.

Damage is defined as the presence and evolution of cracks and cavities at microscopic, mesoscopic or macroscopic level of materials which result deterioration in mechanical properties and may, eventually, lead to failure [92]. According to the difference in materials and loading conditions, damage may be generally divided into: ductile damage, brittle damage, creep damage, and fatigue damage. Ductile damage, occurring simultaneously with large plastic deformation, is the most typical damage in engineering fracture problems. So, in this work, ductile damage is considered and coupled into an elastic-plastic model.

In the theory of damage mechanics, the deterioration of ductile damaged material is generally assumed to be a process of voids nucleation, growth and coalescence. In the phase of voids nucleation, the micro-defects nucleate and germinate, i.e., the deterioration appears. However, it has no significant effects on load-carrying capability of the material, and the damage effects of the material can be neglected. In the following phase of voids growth, the voids and cracks grow considerably as the plastic strain accumulating. The material properties are strongly affected in this phase and damage effects in mechanical properties cannot be ignored. During last phase of voids coalescence, the coalescence of voids and cracks arise and, eventually, the macroscopic crack if formed. The ductile fracture is induced at the end of this phase. The load-carrying capability of the material is reduced as the accumulation of ductile damage in the last two phases until the material losing its loading capability [93].

The continuum mechanics approach to ductile damage problems can be divided into the following two approaches: micromechanical approach and phenomenological approach. In the first approach, the mechanical effect of damage is represented by the void volume fraction f . Then the plastic constitutive equation of voided material and the evolution law

of the void development are derived by means of micromechanics analysis [94]. The Gurson-Tvergaard-Needleman (GTN) model is the representative model of this approach. The second approach to ductile damage analysis, called phenomenological approach, is based on the CDM theory proposed by Lemaitre [95], who has described the ductile damage phenomenon by a phenomenological model with the combination of continuum mechanics and irreversible thermodynamics. Using the method of local state and internal variables, Lemaitre's model strongly coupled damage and elastoplasticity at the constitutive level. In this paper, Lemaitre's model is used to study the ductile damage of the material.

5.1.2 Material Parameter Identification

In the aspect of parameter identification, inverse techniques have become popular in nowadays. In the past, the hand fitting method and the trial and error method are commonly used in material parameter identification [96]. However, the inverse method has become a promising method since optimization techniques and simulation are widely applied. The inverse method considers parameter identification as an optimization problem and generally gives satisfy results [97]. This method tries to find a set of parameters which yields the simulated responses as closely as possible to the experimental response, in other words, inverse method aims at obtaining the set of material parameters by solving the optimization problem which minimize the difference between simulation results and experiment data.

Commonly, in the process of parameter identification by inverse method, the parameters are driven by optimization algorithm, and the objective function, which measures the difference between the simulated and experimental responses, is repeatedly evaluated by numerical simulation until the computational scheme converges. Since optimization algorithm (or optimization technique), which is used to find the solution to the optimization problem, has influences on the convergence speed and the solution of the optimization problem [98], appropriate algorithm should be chosen in the inverse problem of parameter identification.

Generally, optimization algorithms can be divided into two main categories: non-gradient methods (or called direct search methods), which do not employ gradients and involve only evaluations of the objective function, and gradient-based methods, which require computations of the derivatives of the objective function [99]. Gradient-based methods, generally, require a much smaller number of design cycles to converge to an optimum compared to non-gradient methods. However, only convergence to a local minimum is guaranteed for gradient-based methods, while non-gradient methods are able to find global minimum. In addition, gradient-based methods are limited in cases that the objective function is always differentiable and has continuous derivatives over the design domain due to the requirements of computations of derivatives of the objective function. Non-gradient methods do not have limitations on objective function and their convergence speeds are enhanced significantly with the developments of non-gradient optimization techniques.

In the optimization problem of parameter identification using inverse method, the objective function is, normally, a highly nonlinear function and may possess many local minima [100]. Furthermore, the analytical form of the objective function is not known and the evaluation of the objective function is performed by numerical simulation (generally finite element method simulation), in other words, derivatives of the objective function are not available. Taking above aspects into account, a non-gradient global optimization algorithm is preferred to solve the optimization task in parameter identification.

As above mentioned that the objective function in parameter identification problem is not explicitly known and need to be evaluated by repeatedly running numerical simulations, this brings bout heavy computational burden and makes parameter identification process time-consuming. Therefore, in this chapter, the previously proposed KA-CMA-ES using ARP-EI, which has the best success performance among the investigated KA-CMA-ES algorithms, is applied in inverse method of parameter identification.

5.2 Elastic-Plastic Damage Model

In general, a normal elastic-plastic constitutive model contains the following basic components [92]:

(1) The elastic law, which gives the relation between stress and elastic strain;

(2) The yield criterion, which define the limit of plastic behavior;

(3) The flow rule, which describes the evolution of the plastic strain; and

(4) The hardening law, which characterizes the development of the criterion for subsequent yielding.

Accordingly, the elastic-plastic damage model, which considers damage in elastic-plastic material, contains an evolution law of damage variables in addition to these above four components.

In this work, the commonly used von Mises yield criterion, the Swift strain hardening law and Lemaitre's ductile damage are included in the elastic-plastic damage model. The basic components of the elastic-plastic damage model are detailed in this section. In first place, the basic concept of damage variable is introduced.

5.2.1 Damage Variable

The notion of continuum damage mechanics was firstly proposed by Kachanov [101] when he studied on the brittle creep rupture of metal. The first damage variable with physical significance was given later by Rabotnov [92] who proposed the reduction of the cross-sectional area due to micro-cracking as a suitable measure of the state of internal damage. According to the concepts proposed by these authors, the damage variable, D, can be modeled by effective area reduction:

$$D_n = \frac{S - \tilde{S}}{S}, \tag{5.1}$$

where S is the overall area of the element defined by this normal vector $\boldsymbol{n}$, and $\tilde{S}$ is the effective resisting area. When $D_{\boldsymbol{n}} = 0$ corresponds to the undamaged state, and $D_{\boldsymbol{n}} = D_C$ (D_C is a critical value, $0.2 \le D_C \le 0.8$ for metals) corresponds to the rupture of the element [102].

From Equation (5.1), the damage variable $D_{\boldsymbol{n}}$ is associated with $\boldsymbol{n}$ when cracks and cavities are oriented, which leads to tensorial nature of damage variable. However, if cracks and cavities are equally distributed in all directions, $D_{\boldsymbol{n}}$ does not depend on $\boldsymbol{n}$ and it becomes a scalar D, which is called scalar damage variable. In this paper, the damage state is taken to be isotropic and the scalar damage variable D is used in damage model.

In the case that the state of damage is isotropic, the definition of damage by effective area reduction has allowed to define the so-called effective stress:

$$\tilde{\boldsymbol{\sigma}} = \frac{\boldsymbol{\sigma}}{1-D}, \tag{5.2}$$

where $\tilde{\boldsymbol{\sigma}}$ is the effective stress, D is the scalar damage variable, and $\boldsymbol{\sigma}$ is the second-order Cauchy stress tensor. The effective stress of Equation (5.2) simplifies the damage theory, and is employed in a number of damage problems, including ductile damage especially.

5.2.2 Constitutive Model

Lemaitre's ductile damage model is based on CDM theory and in the framework of thermodynamics of irreversible processes. The constitutive equations are derived from thermodynamic (or state) potentials and the evolution equations are given by dissipation potentials, which are detailed in Lemaitre's original model and CDM theory [91]. The elastic-plastic damage model used in this paper (with von Mises yield criterion, Swift hardening law and Lemaitre's ductile damage) are presented in the following.

To begin with, the strain tensor additive split is given:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p, \tag{5.3}$$

where $\boldsymbol{\varepsilon}$ is the total strain tensor, the tensor $\boldsymbol{\varepsilon}^e$ and $\boldsymbol{\varepsilon}^p$ are known, respectively, as the elastic strain tensor and the plastic strain tensor. Correspondingly, the rate form of the additive decomposition reads,

$$\dot{\boldsymbol{\varepsilon}} = \dot{\boldsymbol{\varepsilon}}^e + \dot{\boldsymbol{\varepsilon}}^p. \tag{5.4}$$

The elastic law (coupled with damage) is expressed as:

$$\boldsymbol{\sigma} = (1-D)\mathbf{E} : \boldsymbol{\varepsilon}^e, \tag{5.5}$$

where $\mathbf{E}$ is the isotropic elasticity tensor, $\boldsymbol{\sigma}$ is stress tensor and D is the scalar damage variable.

Since von Mises yield criterion is used, the yield function for elastic-plastic damage model is given by

$$\Phi = \frac{q}{1-D} - \sigma_y, \tag{5.6}$$

where q is the von Mises equivalent stress and σ_y is the yield stress which is defined by Swift hardening law as:

$$\sigma_y\left(\bar{\varepsilon}^p\right) = A\left(\varepsilon_0 + \bar{\varepsilon}^p\right)^n, \tag{5.7}$$

where A, ε_0 and n are material parameters.

The evolution of plastic strain is expressed by

$$\dot{\boldsymbol{\varepsilon}}^p = \frac{\dot{\gamma}}{(1-D)}\sqrt{\frac{3}{2}}\frac{\mathbf{s}}{\|\mathbf{s}\|}, \tag{5.8}$$

where $\dot{\gamma}$ is the plastic multiplier, $\mathbf{s}$ is the deviatoric stress tensor. Accordingly, the evolution of accumulated (equivalent) plastic strain $\bar{\varepsilon}^p$ is:

$$\dot{\bar{\varepsilon}}^p = \frac{\dot{\gamma}}{1-D}. \tag{5.9}$$

The evolution equation of damage variable is

$$\dot{D} = \frac{\dot{\gamma}}{1-D}\left(\frac{-Y}{r}\right)^s \hat{H}\left(\bar{\varepsilon}^p - \bar{\varepsilon}_D^p\right), \tag{5.10}$$

where r and s are material's damage parameters, $\hat{H}(\cdot)$ is the Heaviside step function, $\bar{\varepsilon}_D^p$ is the damage threshold, and $-Y$ is the so-called damage energy release rate, which is defined as

$$
\begin{aligned}
-Y &= \frac{1}{2(1-D)^2}\boldsymbol{\sigma}:\mathbf{E}^{-1}:\boldsymbol{\sigma} \\
&= \frac{1}{2E(1-D)^2}\left[(1+\nu)\boldsymbol{\sigma}:\boldsymbol{\sigma}-\nu\left(\mathrm{tr}(\boldsymbol{\sigma})\right)^2\right] \\
&= \frac{q^2}{2E(1-D)^2}\left[\frac{2}{3}(1+\nu)+3(1-2\nu)\left(\frac{p}{q}\right)^2\right] \\
&= \frac{q^2}{6G(1-D)^2}+\frac{p^2}{2K(1-D)^2},
\end{aligned}
\tag{5.11}
$$

where $\mathbf{E}$ is Young's modulus, ν is Poisson ratio, G is shear modulus, K is bulk modulus, $\mathrm{tr}(\cdot)$ denotes the trace, q is the von Mises equivalent stress and p is the hydrostatic stress.

The loading /unloading conditions $\Phi \le 0,\ \dot{\gamma} \ge 0,\ \Phi\dot{\gamma} = 0$ must be satisfied.

The equations of the elastic-plastic damage model in this chapter are summarized in Table 5.1.

Table 5.1 Equations of elastic-plastic damage model

Strain tensor additive split	$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p$
Elastic law (coupled with damage)	$\boldsymbol{\sigma} = (1-D)\mathbf{E}:\boldsymbol{\varepsilon}^e$
Yield function	$\Phi = q/(1-D) - \sigma_y$
Strain hardening law (Swift law)	$\sigma_y(\bar{\varepsilon}^p) = A(\varepsilon_0 + \bar{\varepsilon}^p)^n$
Evolution of plastic strain	$\dot{\boldsymbol{\varepsilon}}^p = \frac{\dot{\gamma}}{(1-D)}\sqrt{\frac{3}{2}}\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}$
Evolution of damage	$\dot{D} = \frac{\dot{\gamma}}{1-D}\left(\frac{-Y}{r}\right)^s \hat{H}(\bar{\varepsilon}^p - \bar{\varepsilon}_D^p)$
Loading/unloading conditions	$\Phi \le 0,\ \dot{\gamma} \ge 0,\ \Phi\dot{\gamma} = 0$

5.2.3 Numerical Implementation of the Constitutive Model

The fully implicit elastic predictor/return-mapping scheme is commonly used in the numerical implementation of constitutive model [92]. Given the increment of strain $\Delta\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_n$ corresponding to a pseudo-time increment $[t_n, t_{n+1}]$, and given the variables $\boldsymbol{\sigma}_n$, $\boldsymbol{\varepsilon}_n$, $\boldsymbol{\varepsilon}_n^p$, $\bar{\varepsilon}_n^p$ and D_n at t_n, the goal of the algorithm is to find the update values of $\boldsymbol{\sigma}_{n+1}$, $\boldsymbol{\varepsilon}_{n+1}$, $\boldsymbol{\varepsilon}_{n+1}^p$, $\bar{\varepsilon}_{n+1}^p$ and D_{n+1} at t_{n+1}.

The elastic predictor/return-mapping algorithm starts with the so-called elastic trial step. In the elastic trial step, we assume that the step $[t_n, t_{n+1}]$ is elastic; hence, neither damage nor strain hardening evolution takes place at this stage. Therefore, the elastic trial solution is given by:

$$\begin{aligned}
\boldsymbol{\varepsilon}_{n+1}^{e\,\mathrm{trial}} &= \boldsymbol{\varepsilon}_n^e + \Delta\boldsymbol{\varepsilon}, \\
\boldsymbol{\varepsilon}_{n+1}^{p\,\mathrm{trial}} &= \boldsymbol{\varepsilon}_n^p, \\
\bar{\varepsilon}_{n+1}^{p\,\mathrm{trial}} &= \bar{\varepsilon}_n^p, \\
D_{n+1}^{\mathrm{trial}} &= D_n.
\end{aligned} \tag{5.12}$$

The corresponding trial stress can be evaluated by the deviatoric/hydrostatic split of the stress tensor

$$\boldsymbol{\sigma}_{n+1}^{\mathrm{trial}} = \boldsymbol{s}_{n+1}^{\mathrm{trial}} + p_{n+1}^{\mathrm{trial}}\mathbf{I}, \tag{5.13}$$

with

$$\begin{aligned}
\boldsymbol{s}_{n+1}^{\mathrm{trial}} &= \left(1-D_{n+1}^{\mathrm{trial}}\right)2G\varepsilon_{\mathrm{d}\,n+1}^{e\,\mathrm{trial}} = \left(1-D_{n+1}^{\mathrm{trial}}\right)\tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}} = \left(1-D_n\right)\tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}}, \\
p_{n+1}^{\mathrm{trial}} &= \left(1-D_{n+1}^{\mathrm{trial}}\right)K\varepsilon_{\mathrm{v}\,n+1}^{e\,\mathrm{trial}} = \left(1-D_{n+1}^{\mathrm{trial}}\right)\tilde{p}_{n+1}^{\mathrm{trial}} = \left(1-D_n\right)\tilde{p}_{n+1}^{\mathrm{trial}},
\end{aligned} \tag{5.14}$$

where $\tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}} = 2G\varepsilon_{\mathrm{d}\,n+1}^{e\,\mathrm{trial}}$ and $\tilde{p}_{n+1}^{\mathrm{trial}} = K\varepsilon_{\mathrm{v}\,n+1}^{e\,\mathrm{trial}}$ are the effective trial deviatoric and hydrostatic stresses, respectively, $\mathbf{I}$ is the identity tensor, $\varepsilon_{\mathrm{d}\,n+1}^{e\,\mathrm{trial}}$ and $\varepsilon_{\mathrm{v}\,n+1}^{e\,\mathrm{trial}}$ are deviatoric part and volumetric part of the elastic trial strain $\boldsymbol{\varepsilon}_{n+1}^{e\,\mathrm{trial}}$, defined as:

$$\begin{aligned}
\varepsilon_{\mathrm{v}\,n+1}^{e\,\mathrm{trial}} &= \mathrm{tr}\left(\boldsymbol{\varepsilon}_{n+1}^{e\,\mathrm{trial}}\right), \\
\boldsymbol{\varepsilon}_{\mathrm{d}\,n+1}^{e\,\mathrm{trial}} &= \boldsymbol{\varepsilon}_{n+1}^{e\,\mathrm{trial}} - \frac{1}{3}\varepsilon_{\mathrm{v}\,n+1}^{e\,\mathrm{trial}}\mathbf{I}.
\end{aligned} \tag{5.15}$$

With Equation (14), the elastic trial von Mises equivalent stress can be computed:

$$q_{n+1}^{\mathrm{trial}} = \sqrt{3J_2\left(\boldsymbol{s}_{n+1}^{\mathrm{trial}}\right)} = \sqrt{3J_2\left(\left(1-D_{n+1}^{\mathrm{trial}}\right)\tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}}\right)} = \left(1-D_{n+1}^{\mathrm{trial}}\right)\tilde{q}_{n+1}^{\mathrm{trial}}, \tag{5.16}$$

where $\tilde{q}_{n+1}^{\mathrm{trial}} = \sqrt{\frac{3}{2}\tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}} : \tilde{\boldsymbol{s}}_{n+1}^{\mathrm{trial}}}$ is the effective elastic trial von Mises equivalent stress, which is evaluated from effective trial deviatoric stress. The yield function in the present case is then evaluated as

$$\Phi^{\mathrm{trial}} = \frac{q_{n+1}^{\mathrm{trial}}}{1-D_{n+1}^{\mathrm{trial}}} - \sigma_y\left(\bar{\varepsilon}_{n+1}^{p\,\mathrm{trial}}\right) = \tilde{q}_{n+1}^{\mathrm{trial}} - \sigma_y\left(\bar{\varepsilon}_n^p\right). \tag{5.17}$$

If $\Phi^{\text{trial}} \le 0$ the step $[t_n, t_{n+1}]$ is an elastic step and the elastic trial state coincides with the updated state at t_{n+1}, i.e., we set $(\cdot)_{n+1} = (\cdot)_{n+1}^{\text{trial}}$. Otherwise, we turn to the return-mapping step.

In return-mapping step, the following nonlinear return-mapping equations need to solve:

$$\begin{cases} \boldsymbol{\varepsilon}_{n+1}^{e} = \boldsymbol{\varepsilon}_{n+1}^{e\,\text{trial}} - \dfrac{\Delta\gamma}{(1-D_{n+1})}\sqrt{\dfrac{3}{2}}\dfrac{\boldsymbol{s}_{n+1}}{\|\boldsymbol{s}_{n+1}\|}, \\ \bar{\varepsilon}_{n+1}^{p} = \bar{\varepsilon}_{n}^{p} + \dfrac{\Delta\gamma}{(1-D_{n+1})}, \\ D_{n+1} = D_n + \dfrac{\Delta\gamma}{1-D_{n+1}}\left(\dfrac{-Y_{n+1}}{r}\right)^{s}\hat{H}\left(\bar{\varepsilon}_{n+1}^{p} - \bar{\varepsilon}_{D}^{p}\right), \\ \dfrac{q_{n+1}}{(1-D_{n+1})} - \sigma_y\left(\bar{\varepsilon}_{n+1}^{p}\right) = 0. \end{cases} \tag{5.18}$$

The above system of nonlinear equations is unattractive for numerical implementation due to its high computational burden. Therefore, some relatively straightforward operations are performed to reduce the system of equations.

To start with, let us consider the deviatoric/volumetric split of the elastic strain of Equation $(5.18)_1$. This gives

$$\begin{aligned} \varepsilon_{\text{v}\,n+1}^{e} &= \varepsilon_{\text{v}\,n+1}^{e\,\text{trial}}, \\ \boldsymbol{\varepsilon}_{\text{d}\,n+1}^{e} &= \boldsymbol{\varepsilon}_{\text{d},\,n+1}^{e\,\text{trial}} - \frac{\Delta\gamma}{(1-D_{n+1})}\sqrt{\frac{3}{2}}\frac{\boldsymbol{s}_{n+1}}{\|\boldsymbol{s}_{n+1}\|}. \end{aligned} \tag{5.19}$$

According to the elastic law, together with Equation (5.19), the hydrostatic and deviatoric stress can be updated as

$$\begin{aligned} p_{n+1} &= (1-D_{n+1})\,\tilde{p}_{n+1}, \\ \boldsymbol{s}_{n+1} &= (1-D_{n+1})\,\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}} - 2G\Delta\gamma\sqrt{\frac{3}{2}}\frac{\boldsymbol{s}_{n+1}}{\|\boldsymbol{s}_{n+1}\|}, \end{aligned} \tag{5.20}$$

where we have defined

$$\begin{aligned} \tilde{p}_{n+1} &= K\varepsilon_{\text{v}\,n+1}^{e} = K\varepsilon_{\text{v}\,n+1}^{e\,\text{trial}} = \tilde{p}_{n+1}^{\text{trial}}, \\ \tilde{\boldsymbol{s}}_{n+1}^{\text{trial}} &= 2G\boldsymbol{\varepsilon}_{\text{d}\,n+1}^{e\,\text{trial}}. \end{aligned} \tag{5.21}$$

From Equation $(5.20)_2$, we can find that $\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}$ and $\boldsymbol{s}_{n+1}$ are collinear. This implies that $\boldsymbol{s}_{n+1}/\|\boldsymbol{s}_{n+1}\| = \tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}/\|\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}\|$, so that we can equivalently rewrite Equation $(5.20)_2$ and simpler the update formula for $\boldsymbol{s}_{n+1}$ by a straightforward manipulation, which gives

$$\boldsymbol{s}_{n+1} = (1-D_{n+1})\,\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}} - 2G\Delta\gamma\sqrt{\frac{3}{2}}\frac{\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}}{\|\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}\|} = \left(1-D_{n+1}-\frac{3G\Delta\gamma}{\tilde{q}_{n+1}^{\text{trial}}}\right)\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}. \tag{5.22}$$

Thus, with the definition of von Mises equivalent stress, we obtain

$$q_{n+1} = (1-D_{n+1})\,\tilde{q}_{n+1}^{\text{trial}} - 3G\Delta\gamma. \tag{5.23}$$

Consequently, the plastic consistency condition becomes

$$\Phi(\Delta\gamma, D_{n+1}) = \tilde{q}_{n+1}^{\text{trial}} - \frac{3G\Delta\gamma}{1-D_{n+1}} - \sigma_y\left(\bar{\varepsilon}_n^p + \frac{\Delta\gamma}{1-D_{n+1}}\right) = 0. \tag{5.24}$$

By introducing Equation (5.23) and $(5.20)_1$ into the definition of the damage energy release rate of Equation (5.11), the Equation $(5.18)_3$ can be expressed as:

$$D_{n+1} - D_n - \frac{\Delta\gamma}{1-D_{n+1}}\left(\frac{-Y_{n+1}(\Delta\gamma, D_{n+1})}{r}\right)^s = 0, \tag{5.25}$$

where

$$-Y_{n+1}(\Delta\gamma, D_{n+1}) = \frac{\left[(1-D_{n+1})\,\tilde{q}_{n+1}^{\text{trial}} - 3G\Delta\gamma\right]^2}{6G(1-D_{n+1})^2} + \frac{(\tilde{p}_{n+1})^2}{2K}. \tag{5.26}$$

Now, the return-mapping equation system (5.18) has been reduced to a system of two scalar equations (5.24) and (5.25), which is written as

$$\begin{cases} f_1 = \tilde{q}_{n+1}^{\text{trial}} - \dfrac{3G\Delta\gamma}{1-D_{n+1}} - \sigma_y\left(\bar{\varepsilon}_n^p + \dfrac{\Delta\gamma}{1-D_{n+1}}\right) = 0, \\ f_2 = D_{n+1} - D_n - \dfrac{\Delta\gamma}{1-D_{n+1}}\left(\dfrac{-Y_{n+1}(\Delta\gamma, D_{n+1})}{r}\right)^s = 0. \end{cases} \tag{5.27}$$

In this equation system only $\Delta\gamma$ and D_{n+1} are unknown, other variables have defined previously. This nonlinear equation system is solved using Newton-Raphson method. There are two types of coupling can be used: strong coupling and weak coupling. The strong coupling solves the two equations simultaneously to obtain the two unknowns $\Delta\gamma$ and D_{n+1}. While, the weak coupling firstly solves the first equation of the system (5.27) in which we

assume that D_{n+1} is equal to D_n, in order to obtain $\Delta\gamma$. Then the second equation is solved to compute the new damage value D_{n+1}.

Table 5.2 Stress update algorithm for elastic-plastic damage model

(i) Elastic predictor

$$\boldsymbol{\varepsilon}_{n+1}^{e\,\text{trial}} = \boldsymbol{\varepsilon}_n^e + \Delta\boldsymbol{\varepsilon}; \qquad \bar{\varepsilon}_{n+1}^{p\,\text{trial}} = \bar{\varepsilon}_n^p$$

$$\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}} = 2G\varepsilon_{\text{d}\,n+1}^{e\,\text{trial}}; \qquad \tilde{p}_{n+1}^{\text{trial}} = K\varepsilon_{\text{v}\,n+1}^{e\,\text{trial}}$$

$$\tilde{q}_{n+1}^{\text{trial}} = \sqrt{\frac{3}{2}\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}} : \tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}}$$

(ii) Check plastic admissibility

If $\Phi^{\text{trial}} = \tilde{q}_{n+1}^{\text{trial}} - \sigma_y\left(\bar{\varepsilon}_n^p\right) \le 0$ Then

Set $(\cdot)_{n+1} = (\cdot)_{n+1}^{\text{trial}}$ and Exit

Else go to (iii)

(iii) Return-mapping (solve the system for the unknowns $\Delta\gamma$ and D_{n+1})

$$\begin{cases} f_1 = \tilde{q}_{n+1}^{\text{trial}} - \dfrac{3G\Delta\gamma}{1-D_{n+1}} - \sigma_y\left(\bar{\varepsilon}_n^p + \dfrac{\Delta\gamma}{1-D_{n+1}}\right) = 0 \\ f_2 = D_{n+1} - D_n - \dfrac{\Delta\gamma}{1-D_{n+1}}\left(\dfrac{-Y_{n+1}\left(\Delta\gamma, D_{n+1}\right)}{r}\right)^s = 0 \end{cases}$$

where $\sigma_y\left(\bar{\varepsilon}^p\right) = A\left(\varepsilon_0 + \bar{\varepsilon}^p\right)^n$

$$-Y_{n+1}\left(\Delta\gamma, D_{n+1}\right) = \frac{\left[\left(1-D_{n+1}\right)\tilde{q}_{n+1}^{\text{trial}} - 3G\Delta\gamma\right]^2}{6G(1-D_{n+1})^2} + \frac{\left(\tilde{p}_{n+1}\right)^2}{2K}$$

(iv) Update state

$$\boldsymbol{s}_{n+1} = \left(1 - D_{n+1} - \frac{\Delta\gamma 3G}{\tilde{q}_{n+1}^{\text{trial}}}\right)\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}$$

$$p_{n+1} = \left(1-D_{n+1}\right)\tilde{p}_{n+1}^{\text{trial}}$$

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{s}_{n+1} + p_{n+1}\mathbf{I}$$

$$\boldsymbol{\varepsilon}_{n+1}^e = \boldsymbol{\varepsilon}_{n+1}^{e\,\text{trial}} - \Delta\gamma\frac{3}{2}\frac{\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}}{\left(1-D_{n+1}\right)\tilde{q}_{n+1}^{\text{trial}}}$$

$$\boldsymbol{\varepsilon}_{n+1}^p = \boldsymbol{\varepsilon}_n^p + \Delta\gamma\frac{3}{2}\frac{\tilde{\boldsymbol{s}}_{n+1}^{\text{trial}}}{\left(1-D_{n+1}\right)\tilde{q}_{n+1}^{\text{trial}}}$$

$$\bar{\varepsilon}_{n+1}^p = \bar{\boldsymbol{\varepsilon}}_n^p + \frac{\Delta\gamma}{\left(1-D_{n+1}\right)}$$

(v) Exit

Even though the strong coupling gives more accurate results, the weak coupling is generally preferred for computation time reasons in real simulation [103]. Hence, in this paper weak coupling is used to solve the return-mapping equation system (5.27). After obtaining $\Delta\gamma$ and D_{n+1}, stress and other state variables can be updated. The complete stress

update algorithm for the numerical implementation of elastic-plastic damage model by means of the fully implicit elastic predictor/return-mapping scheme is summarized in Table 5.2.

The implementation of the fully implicit elastic predictor/return-mapping scheme is performed in ABAQUS^{TM} subroutine VUMAT (a user subroutine to define material behavior) by FORTRAN codes.

5.3 Inverse Method for Parameter Identification

Using numerical techniques to simulate the response of the structure with the given constitutive model and model parameters is called direct problem. While the parameter identification (or calibration), which determines model parameters on the basis of experimental data, is known as inverse problem [104]. The direct and inverse problems are sketched in Figure 5.1.

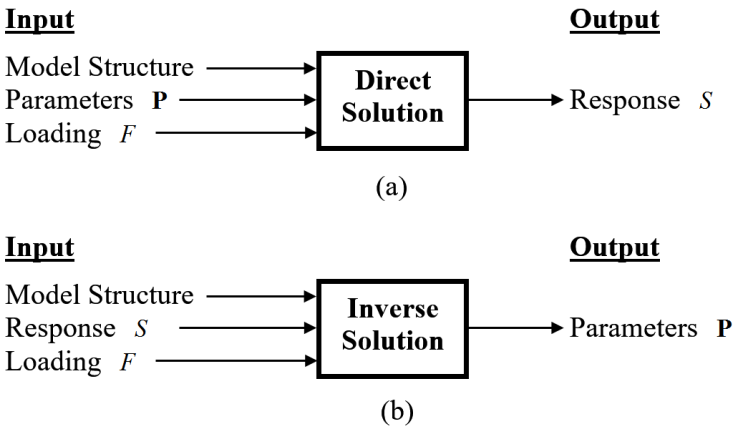

Figure 5.1 Illustration of (a) direct problem and (b) inverse problem

The purpose of parameter identification procedure is to obtain as good agreement as possible between simulated and experimental responses [105, 106]. In other words, the parameter identification process is to find a set of parameters which makes the minimum difference between simulation response and experiment data. Thus, inverse method is a useful tool for determining material parameters [107]. The inverse method of parameter's identification considers the parameter identification as an optimization problem, which is to find a set of parameters that minimize the difference between the experimental data and the

numerical simulations. As an optimization problem, the inverse method of parameter identification basically consists two main parts: the first part is the formulation of the objective function, $f(\mathbf{p})$, which measures the difference between the experimental data and numerical results; another part is the selection of an optimization strategy, which is able to find the minimum of the objective function.

5.3.1 Framework of Inverse Method for Parameter Identification

The general framework of the inverse method for parameter identification is shown in Figure 5.2. A general optimization problem for the inverse method of parameter identification can be expressed as

$$\begin{aligned} &\text{minimize } f(\mathbf{p}) \\ &\text{s.t. } \mathbf{p}_L \le \mathbf{p} \le \mathbf{p}_U \end{aligned} \tag{5.28}$$

where $f(\mathbf{p})$ is the objective function which will be defined in subsequent subsection, $\mathbf{p}_L$ and $\mathbf{p}_U$ are the lower and upper bounds of the parameter vector $\mathbf{p}$.

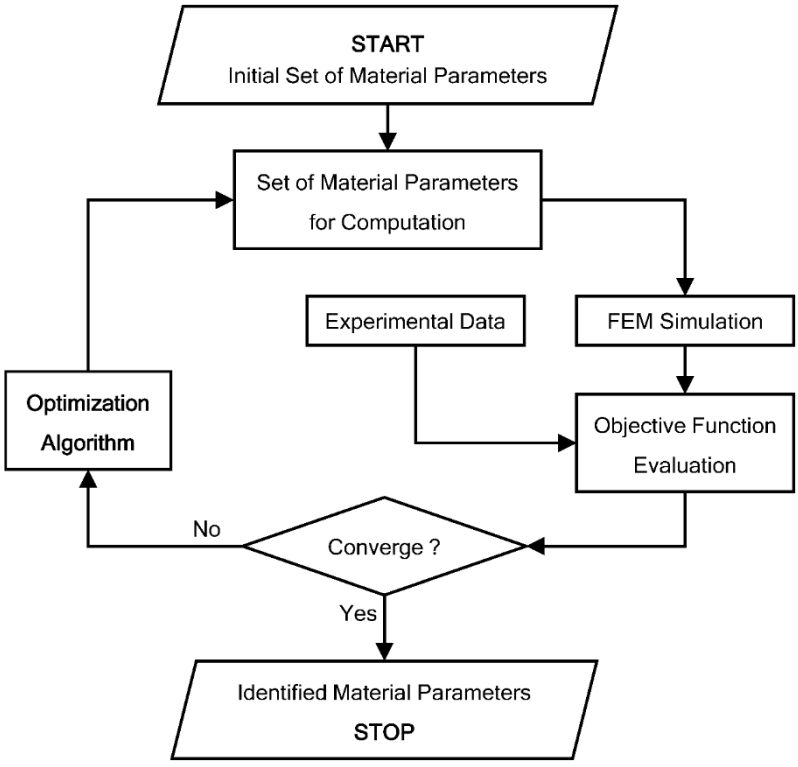

Figure 5.2 Framework of inverse method for parameter identification

Because the objective function is usually implicit and nonlinear, gradient-free global optimization algorithms are commonly used. However, the main drawback of the inverse method is its expensive cost in computation, which is due to the expensive evaluation of the objective function and large number of iterations before convergence. This has promoted the use of the KA-CMA-ES in inverse method.

5.3.2 Objective Function

In this work, our goal is to find a set of material parameters that yields the simulation force-displacement response with minimum difference between the experimental force-displacement curve, as shown in Figure 5.3. The function which provides a scalar measurement of the error between the experimental data and numerical simulation results is chosen as the objective function. We use a formulation based on a least square equation [108], which is expressed as:

$$\begin{aligned} f(\mathbf{p}) &= \frac{\int \left(\mathrm{F}^{\mathrm{sim}}(\mathbf{p}) - \mathrm{F}^{\mathrm{exp}}\right)^2 d\mathrm{D}}{\int \left(\mathrm{F}^{\mathrm{exp}}\right)^2 d\mathrm{D}} \\ &= \frac{\sum_{i=1}^{n} \left[\left(\mathrm{F}_i^{\mathrm{sim}}(\mathbf{p}) - \mathrm{F}_i^{\mathrm{exp}}\right)^2 \left(\mathrm{D}_i - \mathrm{D}_{i-1}\right) \right]}{\sum_{i=1}^{n} \left[\left(\mathrm{F}_i^{\mathrm{exp}}\right)^2 \left(\mathrm{D}_i - \mathrm{D}_{i-1}\right) \right]} \end{aligned} \tag{5.29}$$

where $\mathbf{p}$ is the set of material parameters, $\mathrm{F}_i^{\mathrm{sim}}(\mathbf{p})$ are the numerical simulation responses with the input parameter of $\mathbf{p}$ corresponding to displacement D_i, $\mathrm{F}_i^{\mathrm{exp}}$ are the experimental responses corresponding to D_i, and n is the number of data points. The second line of Equation (5.29) is the discrete formulation of the first line. In evaluation of the objective by its second line (discrete formulation), the simulation and experimental force-displacement data are interpolated at the same query points $(0, \mathrm{D}_1, \mathrm{D}_2, \cdots, \mathrm{D}_n)$ and then $f(\mathbf{p})$ is computed by the discrete formulation.

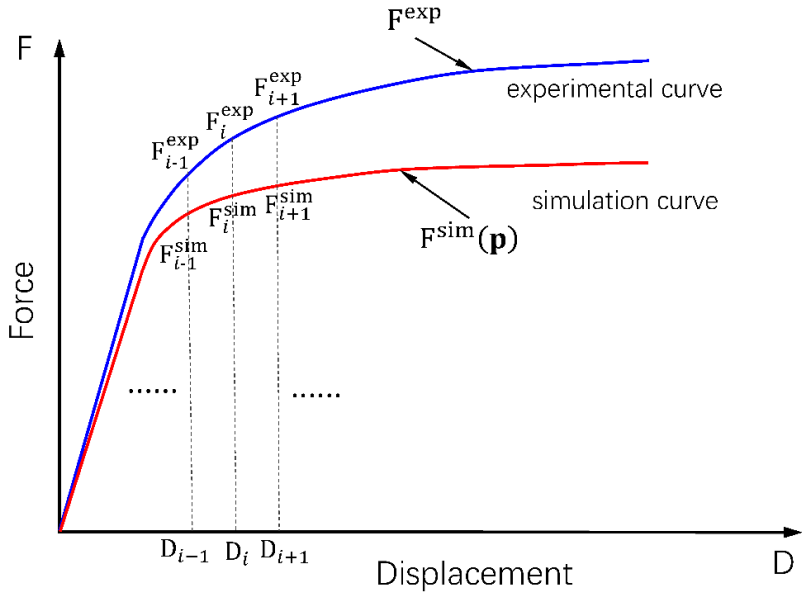

Figure 5.3 Illustration of the difference between experimental and simulation results

In strain hardening parameter identification, the objective function $f(\mathbf{p})$ is computed only on the hardening part, i.e., the integral domain is from $D = 0$ to the necking point. In this situation, the softening part of force-displacement curve is not considered in objective function since damage is not included. When dealing with damage parameters, the softening part of the force-displacement curve needs to be taken into account. In this situation, we need to give precaution on the softening part, in order to build a right objective function.

In the numerical investigation of ductile damage parameters identification by Roux et al. [108], this problem has been solved by adapting the integration domain of objective function in Equation (5.29). Let $D^{sim}_{fracture}$ represents the simulation fracture displacement value and $D^{exp}_{fracture}$ denotes the experimental fracture displacement. Two particular cases, $D^{sim}_{fracture} > D^{exp}_{fracture}$ and $D^{sim}_{fracture} < D^{exp}_{fracture}$, have to be deal with. For both case, the objective function is evaluated between $D = 0$ and $D_{max} = \max\left(D^{sim}_{fracture}, D^{exp}_{fracture}\right)$. The force-displacement data are adapted as following:

- If $D^{sim}_{fracture} > D^{exp}_{fracture}$, experimental data are completed with one new point defined as: the breaking force (load) remains the same whereas the displacement value is set as $D_{max} = \max\left(D^{sim}_{fracture}, D^{exp}_{fracture}\right)$, which is illustrated by the green line in the left of Figure 5.4.

- If $D^{sim}_{fracture} < D^{exp}_{fracture}$, simulation data are completed with one new point defined as: the breaking force (load) remains the same whereas the displacement value is set as $D_{max} = \max\left(D^{sim}_{fracture}, D^{exp}_{fracture}\right)$. This is illustrated by the green line in the right of Figure 5.4.

With above adaptation of force-displacement data, the objective function (Equation (5.29)) can be obtained by evaluating the integration in Equation (5.29) using its discrete formulation.

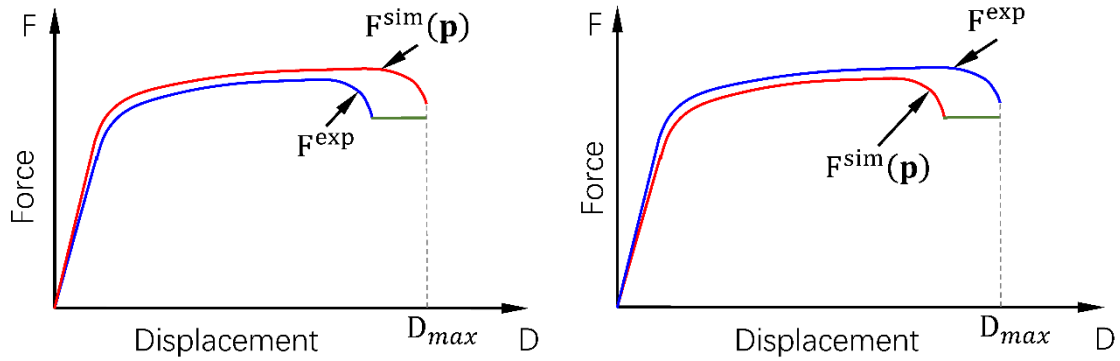

Figure 5.4 Tensile test force-displacement curve, Left: simulation fracture appears for a larger displacement than the experimental fracture displacement; Right: numerical fracture appears for a smaller displacement than the experimental fracture displacement.

5.4 Parameter Identification using Inverse Method

The inverse method presented in the previous section is employed to identify the material parameters that are used in the ductile damage model presented in Section 5.2. The KA-CMA-ES using ARP-EI, which has been proven to be the outstanding algorithm among KA-CMA-ES that have been investigated in previous chapter, is adopted as the optimization algorithm in inverse method of parameter identification. At the same time, the standard CMA-ES is also used, in order to validate the results of inverse method using KA-CMA-ES and evaluate its performance. The objective function, which measures the difference between simulation and experimental response and is minimized in parameter identification, is defined in Equation (5.29). The maximum computational budget is set as 600, i.e., the maximum number of exact objective function evaluations is 600, which is a stopping criterion of the optimization process. Another used termination criterion for optimization is

the tolerance of objective function. We set the tolerance of objective function as $\mathrm{TolFun} = 10^{-6}$. If one of these two criteria is satisfied, the optimization procedure stops. The obtained set of parameters with minimum objective function value are the identified material parameters.

In the ductile damage model of Section 5.2, there are Swift hardening law's parameters (A, ε_0 and n), i.e., the strain hardening parameters, and damage parameters ($\bar{\varepsilon}_D^p$, r, s and D_C) need to be identified. Since in Lemaitre's damage model $s \approx 1.0$ [91], here we set $s = 1.0$. Thus, in this paper, we focus on the identification of strain hardening parameters, A, ε_0 and n, and ductile damage parameters $\bar{\varepsilon}_D^p$, r and D_C. In order to simplify the identification process, these two kinds of parameters (strain hardening parameters and damage parameters) are identified separately. Specifically, the strain hardening parameters (A, ε_0 and n) are identified firstly for the elastoplastic model with Swift hardening law and von Mises yield criterion; then the damage parameters ($\bar{\varepsilon}_D^p$, r and D_C) are identified for the ductile damage model.

The parameter identification is carried out according to the standard tensile test of A 2017-T4. The aluminum 2017-T4 has the density of 2.79×10^3 kg/m^3. Its elastic modulus and Poisson's ratio are: E=72.4 GPa and ν=0.33. The geometry dimensions of the specimens are illustrated in Figure 5.5. The force-displacement curves can be directly obtained from the test system. The first part of the curve is the elastic strain, followed by plastic strain, and then necking and force decrease till the final fracture. The objective of parameter identification is to find an appropriate set of material parameters that yields the best force-displacement response, which has minimum difference with the experimental force-displacement curve.

In parameters identification process, the tensile test process is firstly modeled by Finite Element Method (FEM) simulation using the software ABAQUS. The test specimen is modeled by 3D deformable solid with the element type C3D8R. In the central part, the element size is 1.5 mm, there two layers in the thickness of the specimen. The FEM model of the tensile specimen is shown in Figure 5.6. The loads are applied at the two sides of the

specimen. The left side is fixed in X direction, and the displacement load is applied at the right side.

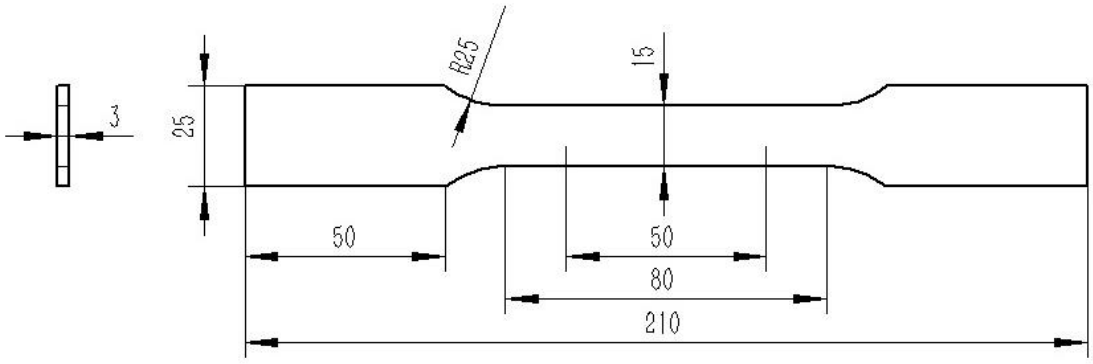

Figure 5.5 Geometry dimension of the test specimen

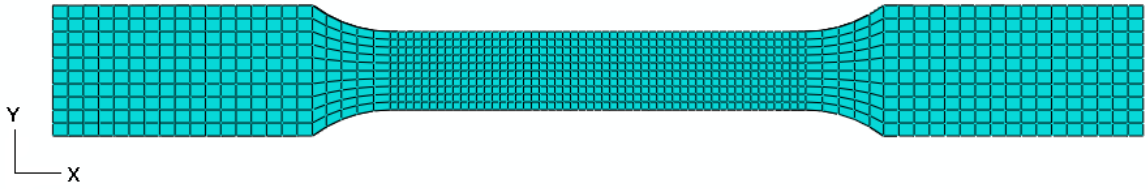

Figure 5.6 FEM model of tensile specimen

5.4.1 Strain Hardening Parameters Identification

The Swift hardening law, which is widely used to model the strain hardening of metallic material, is used to describe the strain hardening behavior of the material. The yield stress defined by Swift hardening law is expressed in Equation (5.7). Previously presented inverse method is used to identify the parameters, A, ε_0 and n, in Swift law.

The inverse parameter identification of strain hardening parameters are independently carried out with the standard CMA-ES and KA-CMA-ES using ARP-EI. In the identification process, FEM simulations of tensile test are repeatedly performed to evaluate the objective function. The lower and upper bounds for the parameters in the identification process and the identified strain hardening parameters are listed in Table 5.3. The convergence graphs and iterative history of the parameters are illustrated in Figure 5.7. The force-displacement

curves corresponding to identified strain hardening parameters and experimental curve are shown in Figure 5.8.

Table 5.3 Results and computational costs of strain hardening parameter identification

Parameter	A	ε_0	n	FES
Lower Bound	600	0	0.1	–
Upper Bound	900	0.03	0.4	–
The Standard CMA-ES	754.1402	0.0087	0.2264	533
KA-CMA-ES using ARP-EI	754.2124	0.0087	0.2264	211

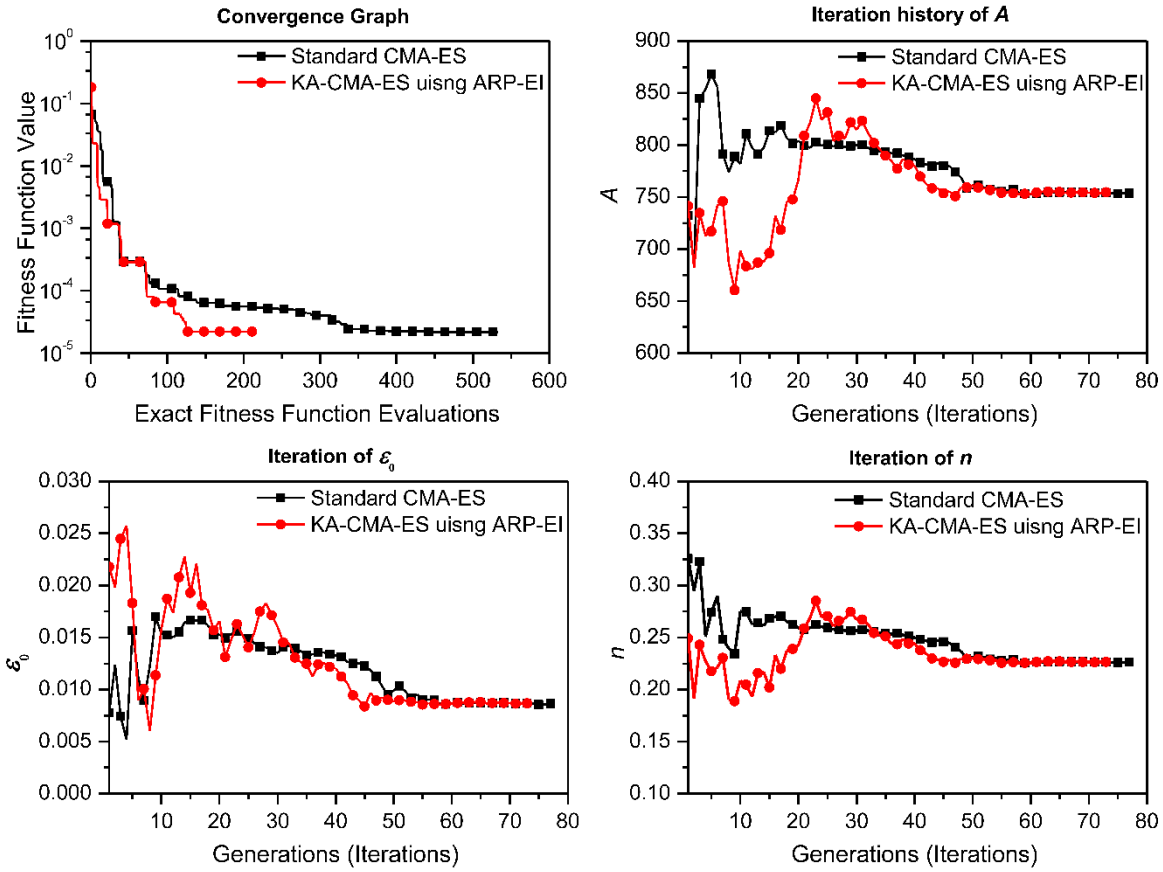

Figure 5.7 Convergence and iteration graphs of strain hardening parameter identification

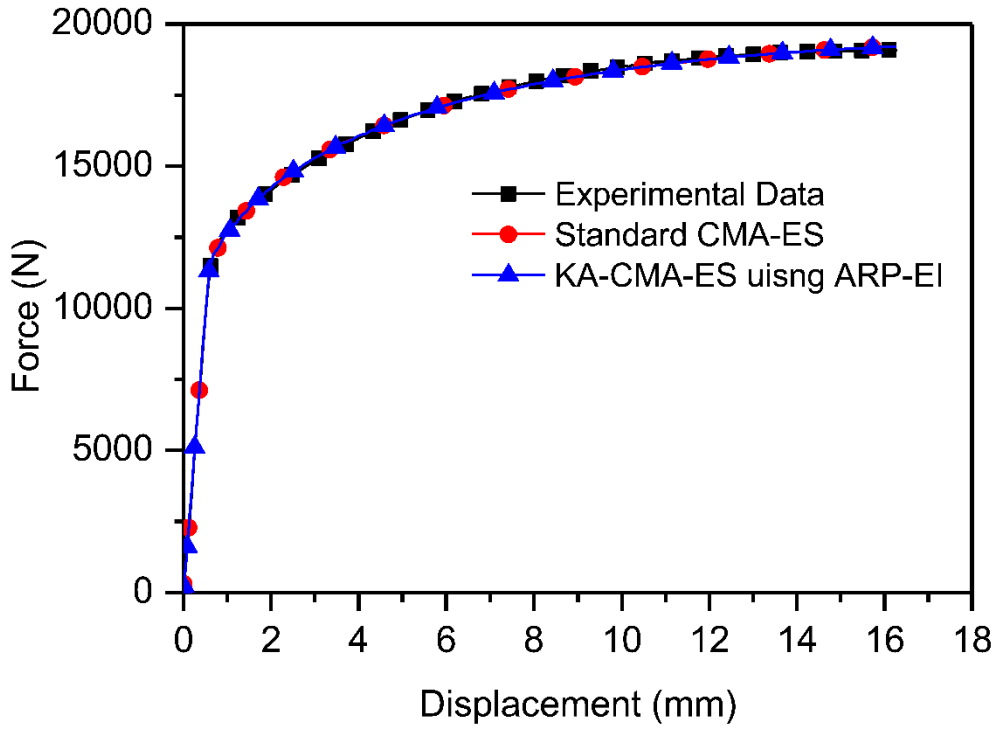

Figure 5.8 Force-displacement curves (before necking) corresponding to identified strain hardening parameters and experimental data

Firstly, in Figure 5.8, it can be observed that the simulation results using the identified parameters are very close to the experimental data. Consequently, Swift's hardening laws with the identified parameters accurately describe the strain hardening behavior of the material. Furthermore, from Table 5.3, it can be seen that the parameter identification results from the inverse method using CMA-ES and those from the inverse method using KA-CMA-ES (ARP-EI) are almost the same. This can also be seen in the iteration history of A, ε_0 and n in Figure 5.7, where these three parameters of the identification process using CMA-ES and KA-CMA-ES converge at almost the same values. In other words, the inverse method using KA-CMA-ES gives a reliable solution for parameter identification problems.

The numbers of objective function evaluations (FES) of the identification process using CMA-ES and KA-CMA-ES are presented in Table 5.3. The identification process using the standard CMA-ES requires 533 objective function evaluations (533 FEM simulations of tensile test with different parameters); whereas, only 211 function evaluations are required in identification using KA-CMA-ES. Obviously, the computational cost of inverse parameter identification using KA-CMA-ES is significantly lower than that of identification

using the standard CMA-ES. This can be found in the convergence graph in Figure 5.7, too. The identification using KA-CMA-ES converges faster than that using the standard CMA-ES. Therefore, efficiency of the parameter identification process has been enhanced by using the proposed KA-CMA-ES.

5.4.2 Damage Parameters Identification

After the strain hardening parameters are identified in previous subsection, we turn to the identification of ductile damage parameters, $\bar{\varepsilon}_D^p$, r and D_C. In the ductile damage model, von Mises yield criterion and Swift hardening law are used. The above identified parameters of Swift hardening law are used here. The lower and upper bounds for the parameters in the identification process and the identified strain damage parameters are listed in Table 5.4. The convergence graphs and iterative history of the damage parameters are illustrated in Figure 5.9. The force-displacement curves corresponding to identified elastic-plastic damage parameters and experimental curve are shown in Figure 5.10.

From the results of damage parameter identification in Table 5.4, the identified damage parameters from inverse method using CMA-ES and KA-CMA-ES are very closed. This additionally indicates the KA-CMA-ES gives reliable results. The iteration histories of damage parameters in Figure 5.9 also demonstrate this.

In Figure 5.10, the force-displacement curve from the numerical simulation using the presented elastic-plastic damage model with the identified parameters is consistent with that from the tensile test. Additionally, the simulation results of the elastic-plastic model and elastic-plastic damage model, and the experimental data are also plotted in Figure 5.11. Obviously, since damage effect is not considered in the elastic-plastic model, the difference between the experimental response and the elastic-plastic model results is significant after the occurrence of necking. In conclusion, the presented elastic-plastic damage model is capable of, and appropriate for, modeling material behavior with consideration of strain hardening and damage effect, and the proposed KA-CMA-ES using ARP-EI is reliable and efficient in inverse method of parameter identification.

Table 5.4 Results and computational costs of damage parameter identification

Parameter	$\bar{\varepsilon}_D^p$	r	D_C	FES
Lower Bound	0.1	2	0	–
Upper Bound	0.18	10	0.8	–
The Standard CMA-ES	0.1306	6.1575	0.0874	295
KA-CMA-ES using ARP-EI	0.1303	6.1583	0.0898	167

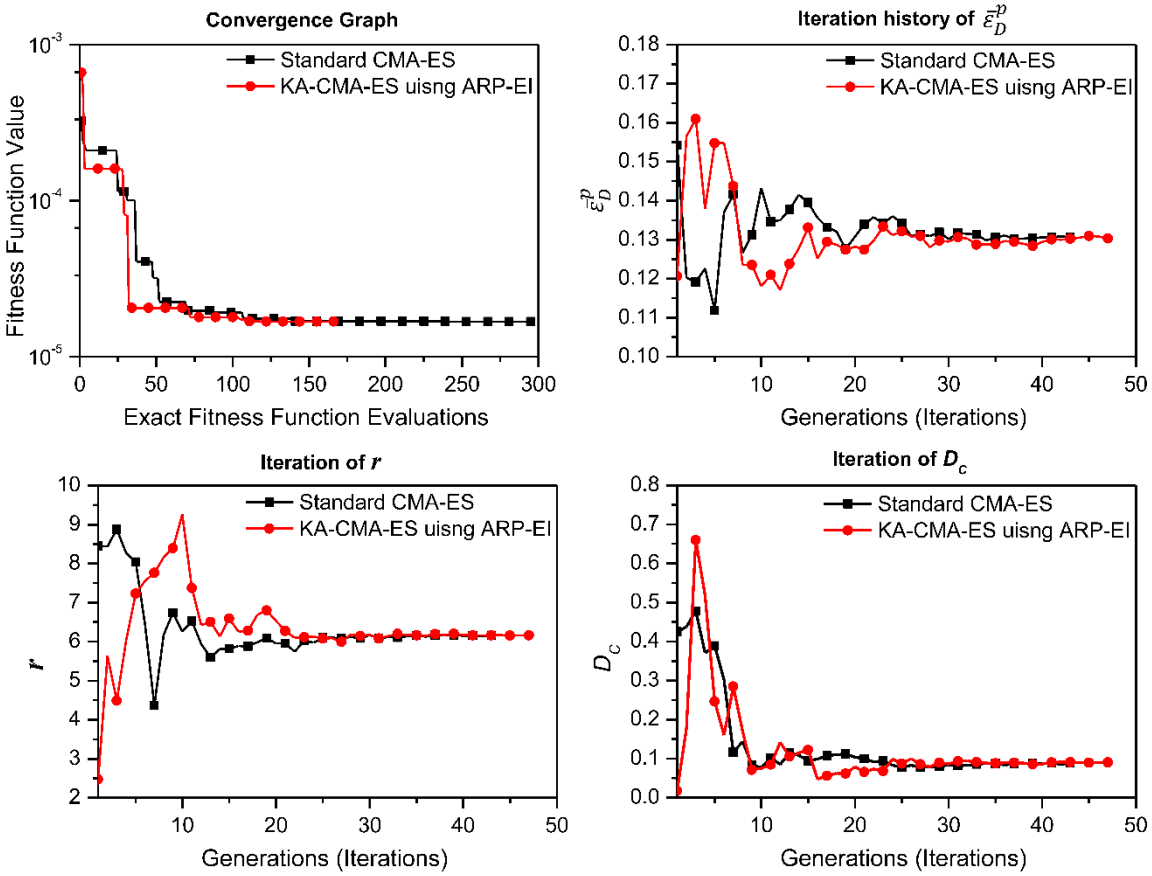

Figure 5.9 Convergence and iteration graphs of damage parameter identification

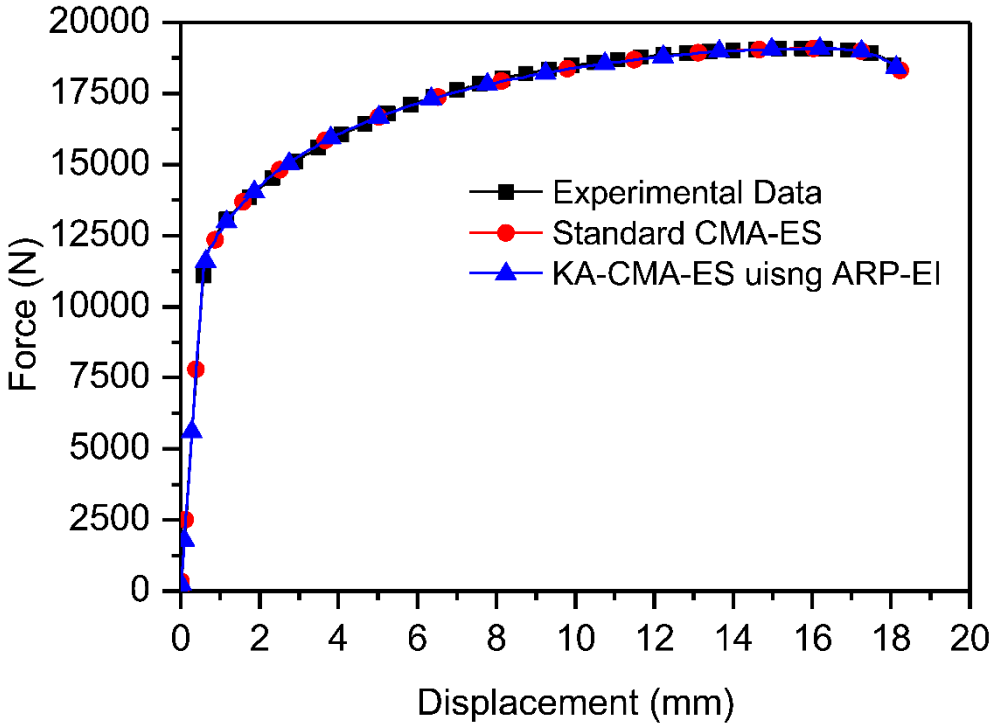

Figure 5.10 Force-displacement curves corresponding to identified damage parameters and experimental data.

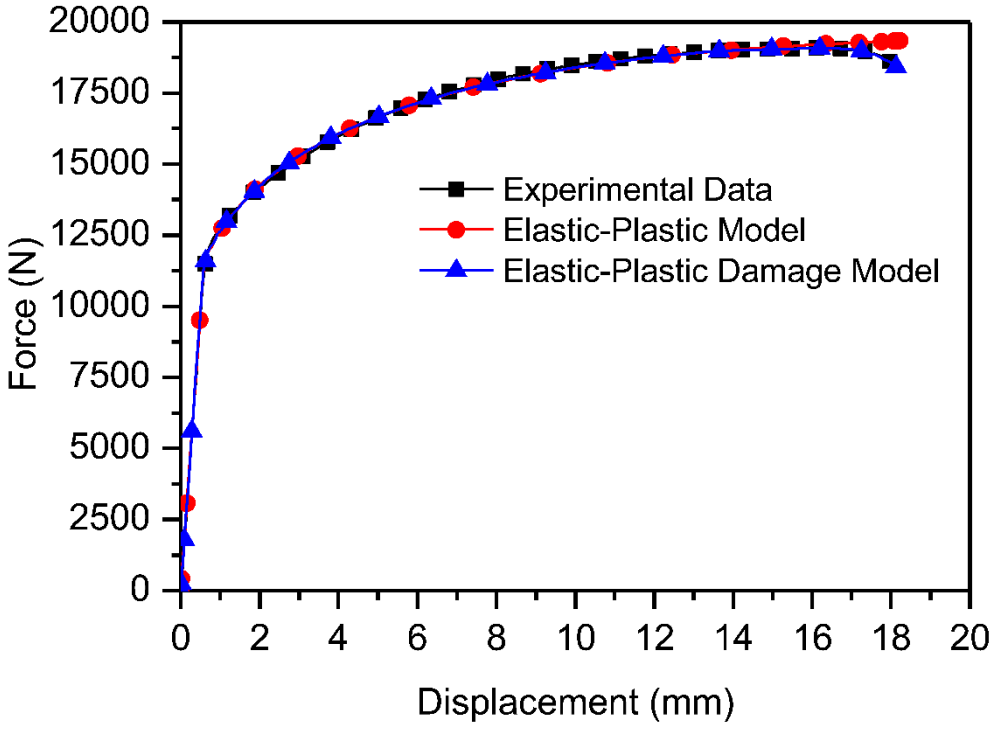

Figure 5.11 Comparison of force-displacement curves of elastic-plastic and elastic-plastic damage models with the identified parameters.

5.5 Summary

This chapter presents a ductile damage model and applies a previously proposed KA-CMA-ES algorithm (KA-CMA-ES using ARP-EI) in inverse method of parameter identification. The ductile damage model, which combines the von Mises yield criterion, Swift's hardening law and Lemaitre's damage model, is implemented by the fully implicit elastic predictor/return-mapping scheme in ABAQUS through the subroutine VUMAT. The numerical implementation algorithm is comprehensively detailed in this chapter. In order to improve the efficiency of the inverse method for parameter identification, previously proposed KA-CMA-ES (ARP-EI) algorithm is used. The proposed ductile damage model is employed for A 2017-T4 and the inverse method using ARP-EI is applied to identify the material parameters based on standard tensile test data.

The results show that Swift's law is adequate to describe the strain hardening behavior of A 2017-T4. With the incorporation of Lemaitre's ductile damage into the elastoplastic model, ductile damage effect can be modeled. The mechanical behavior of A 2017-T4 under tension is accurately model by the presented elastic-plastic damage model.

Applications of the KA-CMA-ES algorithm prove that this algorithm is useful and promising for parameter identification. It enhances the efficiency of the parameter identification process and also gives reliable results.

6. Résumé de la Thèse en Français

6.1 Introduction Générale

L'optimisation est largement demandée et appliquée en science de l'ingénieur. Motivées par les demandes industrielles et de la recherche scientifique, beaucoup de techniques d'optimisation ont été développées. L'algorithme choisi pour un problème d'optimisation dépend en grande partie du type de problème, de la qualité de la solution souhaitée, des ressources informatiques disponibles, du temps CPU et de l'expertise des décideurs.

6.1.1 Motivation

Une problème d'optimisation est dit difficile s'il est coûteux en temps CPU. Ce coût est généralement dû:

- La fonction objectif est évaluée sur la base d'une simulation numérique, qui peut être coûteuse (requiert de quelques minutes à des heures voire des jours de temps de calcul pour chaque cycle de simulation).
- Il n'existe pas d'expression analytique explicite pour la fonction objectif ou ses dérivées. Ainsi, les algorithmes sans dérivation sont nécessaires pour résoudre ce type de problème d'optimisation. Ces derniers nécessitent plusieurs évaluations de la fonction objectif que celles utilisant les dérivés.
- Pour de nombreux problèmes d'optimisation de CAO, la nature de la fonction objectif peut être non lisse, multimodale, discontinue et mal conditionnée. Ces difficultés entraînent un coût de calcul élevé pour trouver l'optimum.

Nous nous concentrons sur les problèmes d'optimisation coûteux sous contraintes décrits comme suit :

$$\begin{aligned} \text{Min} \quad & f(\mathbf{x}) \\ \text{s.c.} \quad & \mathbf{l} \le \mathbf{x} \le \mathbf{u} \end{aligned}$$

où $f(\mathbf{x}): \mathbb{R}^d \to \mathbb{R}$ est la fonction objectif qui est évaluée en exécutant la simulation numérique, le vecteur $\mathbf{x} = [x_1, \cdots, x_d]^T$ représente les d variables de conception, les vecteurs $\mathbf{l}$ et $\mathbf{u}$ sont les limites inférieure et supérieure, respectivement. Les contraintes limitent l'espace de recherche à $[l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d]$.

En raison de la forme implicite de la fonction objectif, les algorithmes évolutifs (EAs), qui sont une classe d'optimiseurs globaux sans dérivées et puissants, sont appropriés pour résoudre des problèmes d'optimisation coûteux. Cependant, la principale difficulté à utiliser les EAs est que ces derniers ont habituellement besoin d'un grand nombre d'évaluation de la fonction objectif avant d'obtenir un résultat satisfaisant. Par conséquent, les EAs assistées par modèle de substitution ont été motivés par la réduction des coûts du temps de calcul.

6.1.2 Objectif

L'objectif de cette thèse est de développer des algorithmes d'optimisation puissants qui peuvent traiter plus efficacement les problèmes d'optimisation coûteux.

Ce travail se concentre sur la stratégie d'évolution (ES) assistée par modèle de substitution pour des problèmes d'optimisation coûteux. La stratégie d'évolution avec adaptation de matrice de covariance (CMA-ES) et le modèle de krigeage sont choisis comme les deux composantes de la stratégie d'évolution assistée par modèle de substitution. Cette CMA-ES assistée par le modèle de krigeage est abrégée par KA-CMA-ES dans ce manuscrit. Notre objectif est d'étudier la CMA-ES existante et de développer de nouveaux algorithmes efficaces de KA-CMA-ES pour des problèmes coûteux.

6.2 Organisation de la Thèse

6.2.1 Premier Chapitre: État de l'Art des Techniques d'Optimisation

Ce chapitre présente un bref aperçu des techniques d'optimisation et de l'état de l'art de l'algorithme de l'évolution assistée par modèle substitutif. On commence par introduire les concepts d'optimisation. Ensuite, on donne un aperçu des algorithmes d'optimisation, y compris les algorithmes avec dérivées et ceux sans dérivées. Enfin, nous présentons l'algorithme de l'évolution assistée par modèle substitutif. Nous examinons le développement d'algorithmes évolutifs (EAs) assistés par substitution. Dans l'optimisation évolutive assistée par substitution, la fonction objectif est remplacée par des modèles de substitution. Les mécanismes d'incorporation des substituts dans les EAs peuvent être divisés en méthodes de remplacement direct et indirect, c'est-à-dire les styles directs et indirects, comme le montre la Figure 6.1. Cette figure montre le panorama de l'optimisation évolutionnaire assistée par modèle substitutif.

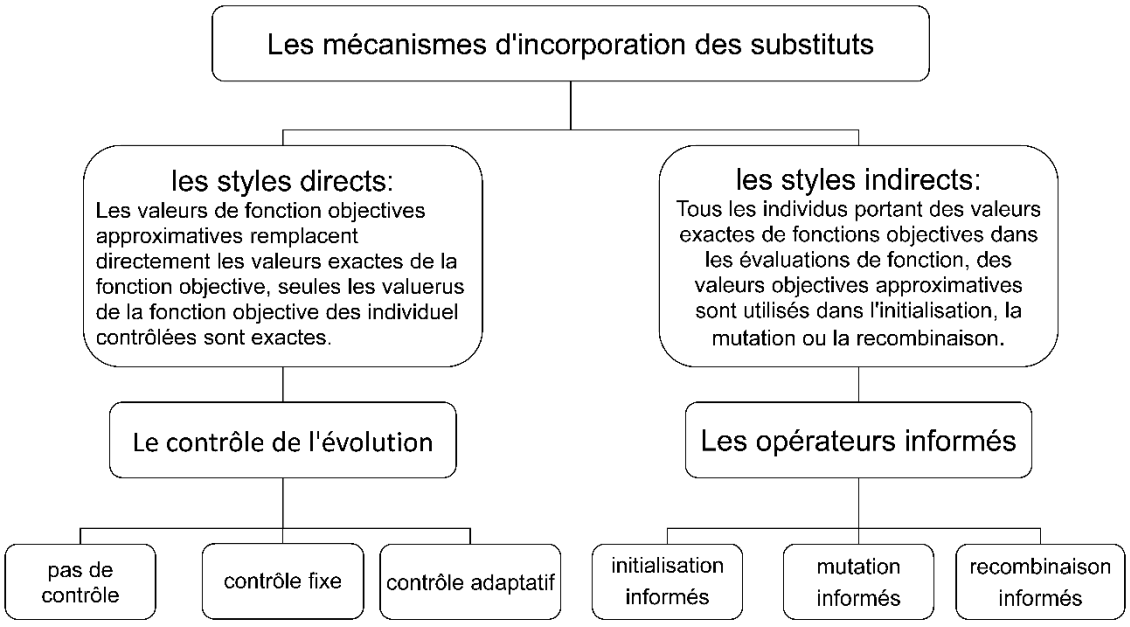

Figure 6.1 Les mécanismes d'incorporation des modèles de substitution dans les EAs

6.2.2 Deuxième Chapitre: Les Stratégies d'Evolution

Ce chapitre donne une description complète des stratégies d'évolution (ES). Une brève introduction des ES est fournie en premier lieu. Ensuite, les principes et les opérateurs évolutifs utilisés dans les ES, c'est-à-dire la sélection, la recombinaison et la mutation, sont décrits dans ce chapitre. Par la suite, nous présentons trois algorithmes typiques de ES, (1+1)-ES, $(\mu/\mu_I,\lambda)$-ES avec adaptation cumulative de taille d'échelon et stratégie d'évolution avec l'adaptation de matrice de covariance (CMA-ES).

Algorithme 6.1 Le $(\mu/\mu_w,\lambda)$-CMA-ES

1: **Données:** $d\in\mathbb{N}_+$, $\lambda=4+\lfloor 3\ln(d)\rfloor$, $\mu=\lfloor\lambda/2\rfloor$,

$$w_i=\frac{\ln(\lambda/2+1/2)-\ln(i)}{\sum_{j=1}^{\mu}\left[\ln(\lambda/2+1/2)-\ln(j)\right]}\ \text{for } i=1,\cdots,\mu,\ \ \mu_w=\left(\sum_{i=1}^{\mu}w_i\right)^2\Big/\sum_{i=1}^{\mu}w_i^2,$$

$$c_\sigma=\frac{\mu_w+2}{d+\mu_w+5},\ \ d_\sigma=1+2\max\left(0,\sqrt{\frac{\mu_w-1}{d+1}}-1\right)+c_\sigma,$$

$$c_c=\frac{4+\mu_w/d}{d+4+2\mu_w/d},\ \ c_1=\frac{2}{(d+1.3)^2+\mu_w},\ \ c_u=\min\left(1-c_1,2\frac{\mu_w-2+1/\mu_w}{(d+2)^2+\mu_w}\right).$$

2: **initialiser** $\mathbf{m}^{(0)}\in\mathbb{R}^d$, $\sigma^{(0)}>0$, $\mathbf{p}_{\mathrm{c}}^{(0)}=\mathbf{0}$, $\mathbf{p}_\sigma^{(0)}=\mathbf{0}$, $\mathbf{C}^{(0)}=\mathbf{I}$, $g\leftarrow 0$

3: **répéter**

4: **Pour** $k=1,\cdots,\lambda$ **faire**

5: $\mathbf{z}_k=\mathcal{N}_d(\mathbf{0},\mathbf{I})$ //i.i.d. pour chaque $\mathbf{z}_k$

6: $\mathbf{x}_k=\mathbf{m}^{(g)}+\sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{1/2}\mathbf{z}_k$ //mutation

7: $f_k=f(\mathbf{x}_k)$

8: **fin pour**

9: $$\mathbf{m}^{(g+1)}\leftarrow\mathbf{m}^{(g)}+\sigma^{(g)}\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\sum_{i=1}^{\mu}w_i\mathbf{z}_{i:\lambda}$$

10: $$\mathbf{p}_\sigma^{(g+1)}\leftarrow(1-c_\sigma)\mathbf{p}_\sigma^{(g)}+\sqrt{c_\sigma(2-c_\sigma)}\sqrt{\mu_w}\sum_{i=1}^{\mu}w_i\mathbf{z}_{i:\lambda}$$

11: $$\sigma^{(g+1)}\leftarrow\sigma^{(g)}\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{\mathbb{E}\left(\left\|\mathcal{N}_d(\mathbf{0},\mathbf{I})\right\|\right)}-1\right)\right)$$

12: $$\mathbf{p}_{\mathrm{c}}^{(g+1)}\leftarrow(1-c_{\mathrm{c}})\mathbf{p}_{\mathrm{c}}^{(g)}+h_\sigma\sqrt{c_{\mathrm{c}}(2-c_{\mathrm{c}})}\sqrt{\mu_w}\sum_{i=1}^{\mu}w_i\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}$$

13: $$\mathbf{C}^{(g+1)}\leftarrow(1-c_1-c_u)\mathbf{C}^{(g)}+c_1\left(\mathbf{p}_{\mathrm{c}}^{(g+1)}\left(\mathbf{p}_{\mathrm{c}}^{(g+1)}\right)^{\mathrm{T}}+\delta(h_\sigma)\mathbf{C}^{(g)}\right)+c_\mu\sum_{i=1}^{u}w_i\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\left(\left(\mathbf{C}^{(g)}\right)^{\frac{1}{2}}\mathbf{z}_{i:\lambda}\right)^{\mathrm{T}}$$

14: $g\leftarrow g+1$

15: **Test d'arrêt vérifié**

Parmi les ES, la stratégie d'évolution avec l'adaptation de matrice de covariance (CMA-ES) est une stratégie d'évolution très développée et est devenue une norme pour l'optimisation évolutive continue. Il s'agit d'un puissant algorithme d'optimisation et se comporte particulièrement bien dans des problèmes de back-box multimodaux non lisses. Le CMA-ES adopte l'opérateur de mutation corrélée, ce qui en fait un algorithme de haut niveau comparé à d'autres algorithmes qui utilisent la mutation isotrope. Dans CMA-ES, deux techniques, à savoir l'adaptation de la matrice de covariance (CMA) et l'adaptation cumulative des échelons (CSA), sont utilisées pour adapter respectivement la matrice de covariance de la mutation et la taille de l'échelon. Le CMA-ES est choisi comme la stratégie d'évolution de la base pour les ES assistées par substitution dans ce travail, en raison de sa puissance et de son succès dans l'optimisation continue avec des fonctions objectifs implicites. L'algorithme CMA-ES est donné dans l'Algorithme 6.1.

6.2.3 Troisième Chapitre: Modélisation de Substitution

Ce chapitre examine la méthode de modélisation substitutive, qui est une approche pour décrire le comportement (entrée-sortie) du modèle de simulation. Les trois étapes de la modélisation de substitution sont décrites dans ce chapitre : (i) la conception des expériences (DOE) ; (ii) la formation du modèle de substitution et (iii) la validation du modèle.

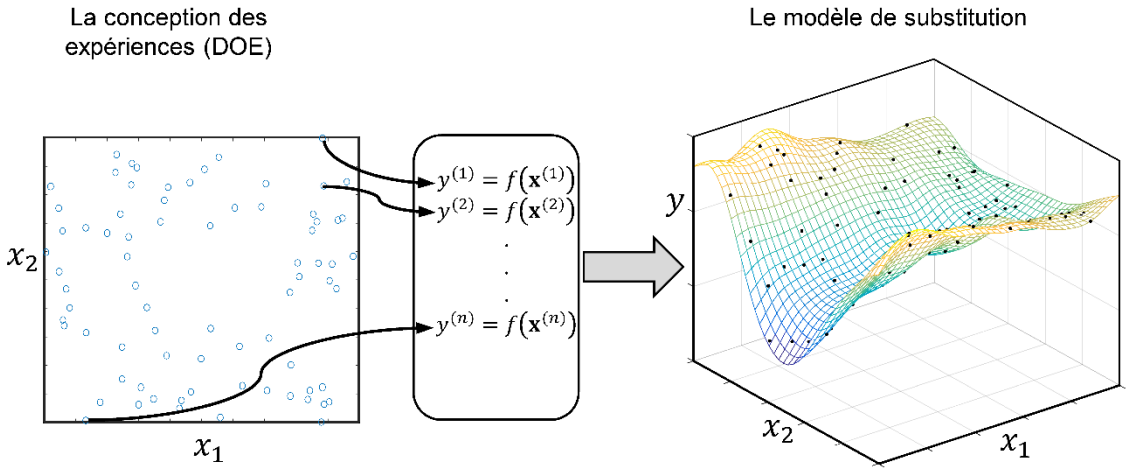

Figure 6.2 Processus de modélisation de substitution

La modélisation substitutive est liée à la construction de modèles mathématiques pour décrire les relations entre les entrées et les sorties spécifiques exposées par le modèle de simulation (ou le système), basée sur un ensemble de données limitées acquises en exécutant le modèle de simulation avec une entrée intelligemment choisie. Le processus de modélisation de substitution est illustré par la Figure 6.2.

Avec un modèle subrogé formé, la sortie des points non testés peut être prédite à moindre coût par le modèle. Par conséquent, les modèles de substitution peuvent être utilisés dans l'optimisation évolutive en remplaçant l'évaluation coûteuse de la fonction de fitness par la prédiction du modèle de substitution moins coûteux que l'évaluation exacte de la condition physique.

En se basant sur l'apprentissage mécanique et les techniques d'apprentissage statistique, jusqu'à présent, plusieurs modèles de substitution ont été utilisés dans le calcul évolutionnaire assisté. Les modèles de substitution les plus populaires, y compris la régression polynomiale, le modèle de krigeage, les fonctions de base radiales, les réseaux de neurones et les machines de vecteurs de soutien.

Modèle de krigeage

Le modèle de krigeage est une méthode de modélisation basée sur le processus de Gauss pour interpoler des données déterministes sans bruit et s'est révélé utile dans une grande variété de domaines. La caractéristique distinctive du modèle de Krigeage est qu'il fournit non seulement une réponse prédite (moyenne de prédiction) à un point non échantillonné, mais aussi une estimation de la variance de prédiction (ou écart type de prédiction). Cette variance donne une indication de l'incertitude dans le modèle de krigeage, qui résulte de la construction de la fonction de covariance. Cette dernière est basée sur l'idée que lorsque les points d'entrée sont proches l'un de l'autre, la corrélation entre leurs sorties correspondantes sera élevée. Par conséquent, l'incertitude associée aux prédictions du modèle sera faible pour les points d'entrée qui sont près des points de formation, et augmentera à mesure que l'on s'éloigne des points de formation. L'écart-type de la prédiction et la moyenne de la prédiction fournissent des informations précieuses pour équilibrer l'exploitation et donc bénéfique pour

le calcul évolutionnaire assisté. Compte tenu de cet avantage, le modèle de krigeage est utilisé comme substitut dans la stratégie d'évolution assistée par substitution.

6.2.4 Quatrième Chapitre: CMA-ES Assistée par Modèle de Krigeage

Ce chapitre se concentre sur les stratégies d'évolution avec l'adaptation de matrice de covariance assistée par le modèle de krigeage (KA-CMA-ES), dans le cadre de l'optimisation évolutive assistée par substitution. Une brève introduction de la stratégie d'évolution assistée par modèle est présentée en premier lieu. Ensuite, nous décrivons les mécanismes d'incorporation du modèle de substitution dans ES. Ensuite, de nouvelles méthodes de sélection d'ensembles d'apprentissage, de présélection, du contrôle d'évolution ont été développées et des algorithmes concrets de KA-CA-ES sont formulés. En outre, nous effectuons des études expérimentales de KA-CMA-ES pour étudier et analyser la performance des algorithmes proposés.

6.2.4.1 Algorithmes de KA-CMA-ES

Le mécanisme d'incorporation du modèle de krigeage dans CMA-ES est la partie essentielle de KA-CMA-ES. Dans cette thèse, on adopte les quatre mécanismes d'incorporation suivants ou les stratégies d'exploitation du modèle : la présélection, le contrôle individuel, le classement approximatif et le contrôle basé sur la génération. Dans le contrôle individuel, différents critères qui sont utilisés pour sélectionner les individus à contrôler et sont appelés métriques dans ce travail, sont étudiés. Les stratégies d'exploitation et de mesure des modèles sont décrites ci-dessous.

6.2.4.2 Présélection

La présélection de solutions prometteuses (basée sur un modèle approximatif) est une stratégie populaire d'exploitation de l'information à partir du modèle de substitution $\hat{f}$ dans les stratégies d'évolution. Dans la stratégie de présélection, $\lambda_{\text{Pre}} > \lambda$ (λ est la taille de la population de ES) les individus sont générés à chaque étape, puis tous les λ_{Pre} individus sont évalués par modèle approximatif $\hat{f}$, après que λ parmi les λ_{Pre} meilleurs individus sont

sélectionnés pour évaluer par la fonction de fitness d'origine f. L'idée de base de cette approche est que seuls les individus les plus prometteurs avec une bonne prédiction de la condition physique sont évalués avec la fonction de remise en forme, ce qui entraîne une réduction du nombre d'appels de remise en forme réels coûteux. La présélection est illustrée à la Figure 6.3.

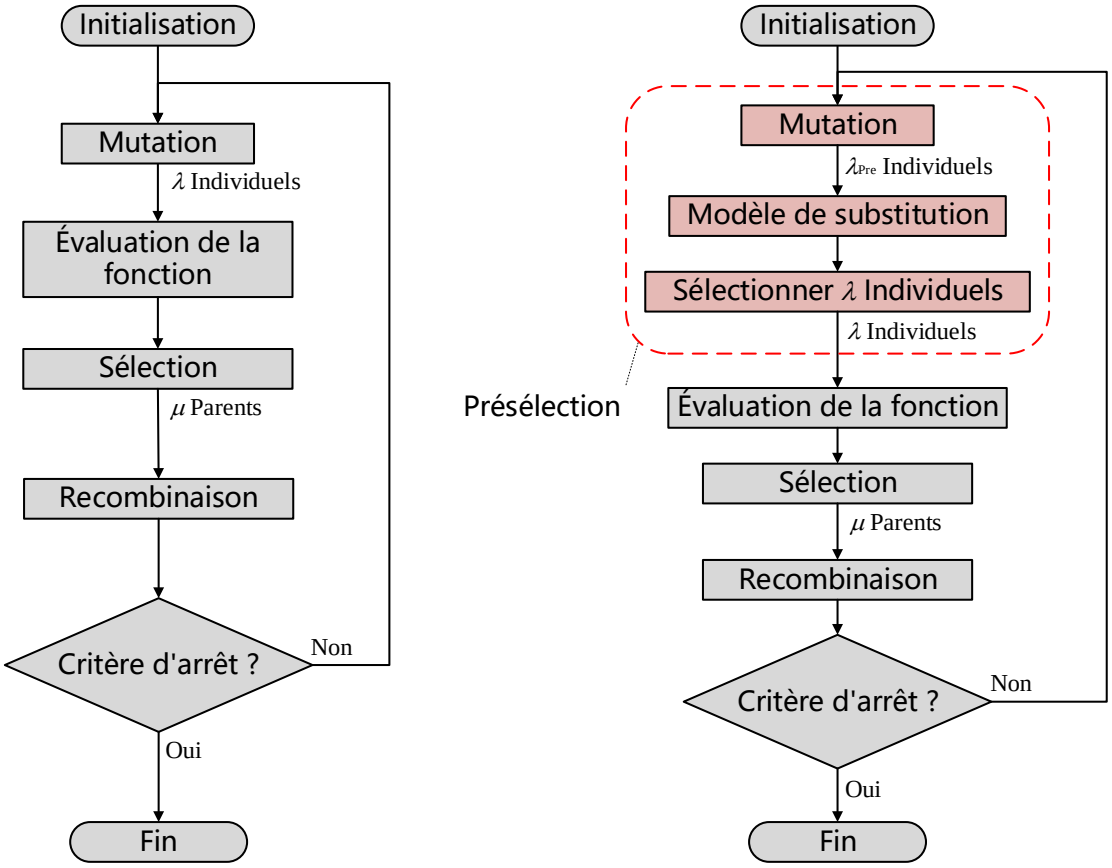

Figure 6.3 Illustration de la présélection dans la stratégie d'évolution assistée par modèle de substitution: (1) la norme (μ,λ)-ES et (2) l'ES assistée par la présélection.

Il existe deux stratégies de présélection : (1) la présélection sans contrôle de l'impact du modèle (PS), et (2) la présélection avec contrôle de l'impact du modèle (CPS) utilisé dans KA-CMA-ES. Dans PS, la taille de la population de présélection λ_{Pre} reste constante pendant le processus d'optimisation évolutive. En CPS, λ_{Pre} a été contrôlé dynamiquement sur la base de la mesure de la qualité du modèle.

6.2.4.3 Contrôle Personnalisé de l'Evolution

Le contrôle de l'évolution est populaire dans les algorithmes évolutifs assistés par substitution, y compris les stratégies d'évolution. Le contrôle de l'évolution signifie que, dans le calcul évolutif assisté par substitution, la fonction d'aptitude initiale est utilisée pour évaluer certains/tous les individus dans certaines/toutes les générations. Un individu qui est évalué à l'aide de la fonction d'aptitude d'origine est appelé un individu contrôlé. De même, une génération dans laquelle tous ses individus sont évalués en utilisant la fonction de fitness d'origine est appelée génération contrôlée. Le contrôle basé sur individuel et le contrôle basé sur la génération sont illustrés à la Figure 6.4.

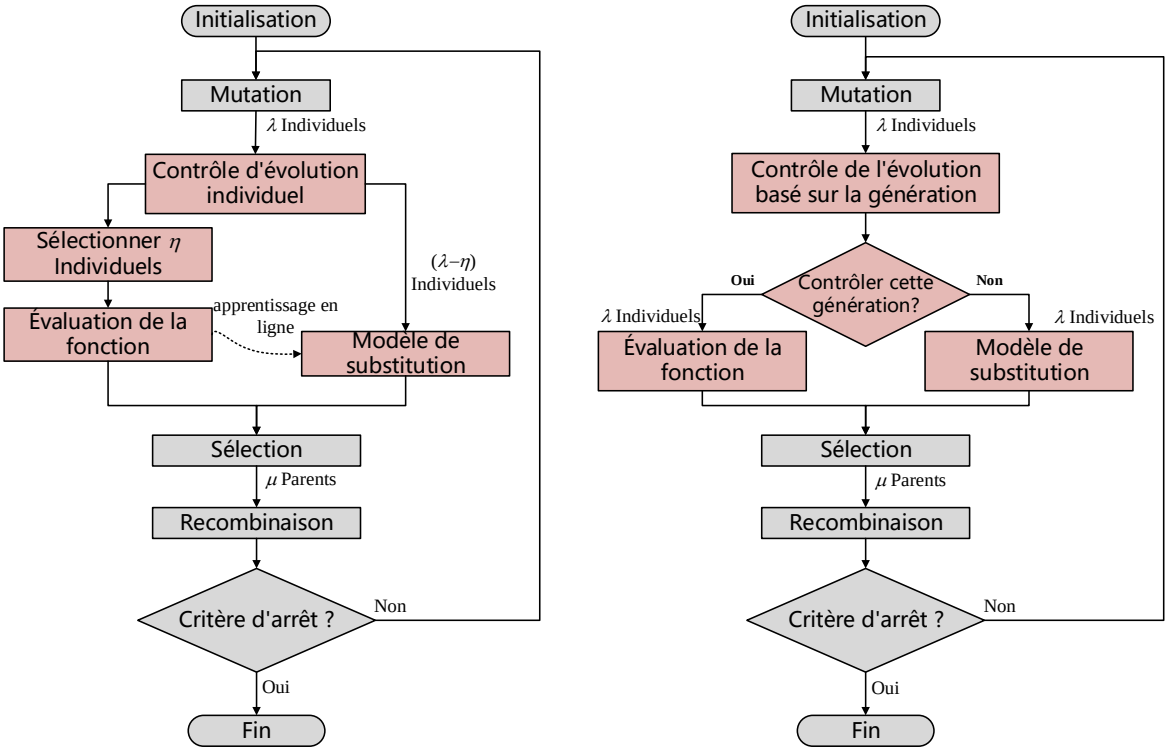

Figure 6.4 Illustration du contrôle de l'évolution dans la stratégie d'évolution assistée par substitution : (1) contrôle basé sur l'individuel et (2) contrôle basé sur la génération.

Dans le contrôle basé sur l'individuel, une partie des individuels dans la population sont choisis et évalués avec la fonction d'objectif. Les individuels contrôlés peuvent être choisis au hasard ou par la stratégie « best ». Si la fréquence du contrôle d'évolution individuel est fixe, c'est-à-dire qu'un nombre fixe d'individus est contrôlé dans chaque génération, on l'appelle contrôle individuel fixe (FIC). Si la fréquence du contrôle de l'évolution dépend de la fidélité du modèle de substitution. Cette stratégie est appelée contrôle de l'évolution adaptative.

Dans le but de tirer les avantages du contrôle d'évolution individuel et adaptative basé sur l'individu, une stratégie de contrôle basée sur des individus appelée contrôle individuel mixte (MIC) qui combine les deux fonctions du contrôle fixe et adaptif d'une certaine manière, est proposée dans ce travail.

6.2.4.4 Procédure du Classement Approximatif

En 2004, Runarsson a proposé la procédure de classement approximatif (ARP) pour évaluer la qualité des modèles de substitution et servir de contrôle de l'évolution dans l'ES assistée. La procédure de classement approximatif évalue la qualité du modèle de substitution par sa cohérence dans le classement de la population plutôt que par sa précision statistique. La procédure de classement approximatif détermine le nombre d'individus contrôlés dans chaque génération de la façon suivante : les individus sont successivement sélectionnés pour être évalués en fonction de leur aptitude approximative et ensuite ajoutés à l'ensemble d'entraînement jusqu'à ce que la sélection de substitution des parents reste inchangée dans deux itérations cycles.

6.2.4.5 Contrôle d'Evolution basé sur la Génération

Dans le contrôle de l'évolution basé sur la génération, tous les individus d'une génération sélectionnée seront évalués par la fonction d'aptitude initiale. La sélection de génération peut être aléatoire ou avec une fréquence fixe. Si la fréquence du contrôle de génération est fixe, on l'appelle contrôle de génération fixe (FGC). Si la fréquence du contrôle de génération dépend de la qualité du modèle de substitution, elle est connue sous le nom de contrôle adaptatif basé sur la génération (AGC).

Dans cette section, il est proposé d'utiliser la qualité du modèle pour déterminer si la prochaine génération est le contrôle ou non. La qualité du modèle de substitution est estimée dans chaque génération contrôlée, si $Q \geq Q^{TS}$ (Q^{TS} est la valeur critique ou le seuil de qualité du modèle), la prochaine génération est évaluée par le modèle de substitution ; sinon, la prochaine génération est contrôlée et évaluée par la fonction de fitness d'origine. Les mesures de la qualité du modèle peuvent être la mesure de qualité de modèle de sélection

proposée à l'aide de poids de recombinaison Q_w, la mesure de qualité du modèle fondée sur la sélection $Q_{\text{selection}}$ proposée par Ulmer et al., et le coefficient de corrélation de rang ρ_{rank}.

Résumé des Algorithmes KA-CMA-ES dans ce Travail

En combinant le modèle de krigeage et CMA-ES à travers la gestion de modèle décrite ci-dessus, les différents algorithmes peuvent être formulés. Quatre groupes d'algorithmes KA-CMA-ES ont été développés :

1) KA-CMA-ES en utilisant la présélection. Ce groupe comprend KA-CMA-ES utilisant PS (présélection sans contrôle d'impact de modèle) et CPS (présélection avec contrôle d'impact de modèle).

2) KA-CMA-ES en utilisant un contrôle individuel. Ce groupe inclut KA-CMA-ES utilisant FIC (contrôle individuel fixe avec métrique) et MIC (le contrôle individuel mixte proposé).

3) KA-CMA-ES en utilisant la procédure de classement approximatif (ARP).

4) KA-CMA-ES utilisant le contrôle basé sur la génération, qui comprend KA-CMA-ES utilisant FGC (contrôle basé sur la génération fixe) et AGC (contrôle adaptatif basé sur la génération).

6.2.4.6 Études Expérimentales de KA-CMA-ES

En utilisant différents modèles d'exploitation dans KA-CMA-ES, différents algorithmes ont été développés. Ceux étudiés dans cette thèse peuvent être divisés en quatre groupes : KA-CMA-ES utilisant la présélection qui comprend KA-CMA-ES utilisant PS et CPS, KA-CMA-ES en utilisant un contrôle individuel qui comprend KA-CMA-ES utilisant FIC et MIC, KA-CMA-ES en utilisant une procédure approximative de classement, et KA-CMA-ES utilisant un contrôle basé sur la génération qui comprend KA-CMA-ES utilisant FGC et AGC. Dans les études expérimentales de KA-CMA-ES, différents aspects des algorithmes KA-CMA-ES ont été examinés en exécutant les expériences sur 40 problèmes de benchmark (dont 12 fonctions de test avec des dimensions différentes). Le taux de succès

(SR), la performance de succès (SP) et/ou la performance d'accélération (SPU) des expériences sont trois mesures pour évaluer et analyser la performance des algorithmes.

6.2.4.7 Résultats Expérimentaux

Dans l'étude de KA-CMA-ES en utilisant la présélection sans contrôle d'impact de modèle (PS), différentes méthodes de sélection d'ensembles d'entraînement, y compris 'Récemment' (les points récemment évalués), 'kNN' Moyenne basée sur la distance de Mahalanobis), et 'Interval' (la méthode proposée d'intervalle de confiance), ont été étudiés. Les résultats sont montrés dans la Figure 6.5. Les résultats de cette étude ont démontré que la méthode d'intervalle de confiance proposée pour la sélection d'ensembles d'apprentissage semble supérieure aux méthodes «Récemment» et «kNN» pour la sélection d'ensembles d'apprentissage. KA-CMA-ES utilisant PS avec 'Interval' a un taux de réussite plus élevé et une meilleure performance de succès que celle utilisant 'Récemment' et 'kNN'. Ainsi, la méthode de l'intervalle de confiance est suggérée pour la sélection des ensembles d'entraînement dans la stratégie d'évolution assistée.

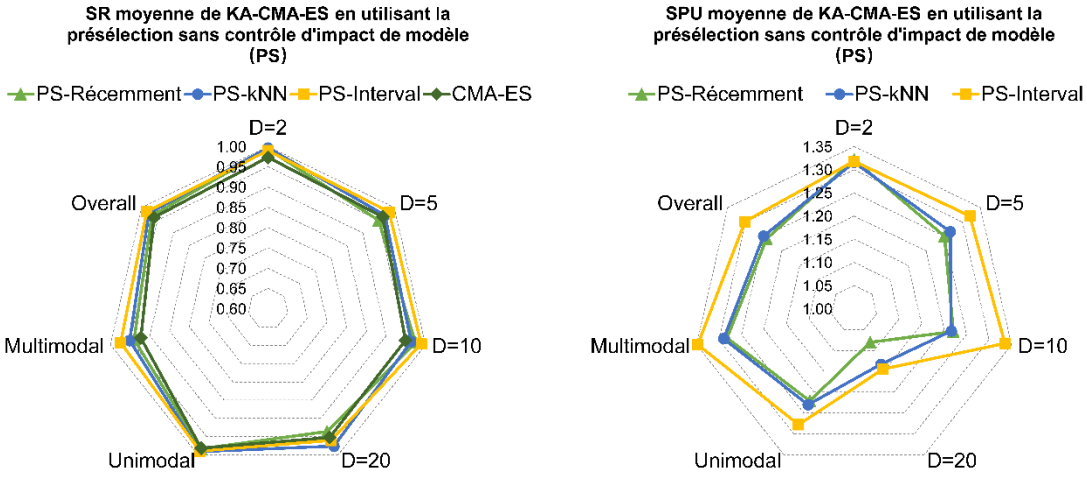

Figure 6.5 Taux de réussite moyen et des performances d'accélération de KA-CMA-ES en utilisant la présélection sans contrôle de l'impact du modèle (PS).

Dans l'étude de KA-CMA-ES en utilisant la présélection avec le modèle de contrôle d'impact (CPS), trois mesures de qualité de modèle Q_w, $Q_{\text{selection}}$ et ρ_{rank}, sont utilisées et comparées. Les résultats (voir Figure 6.6) montrent que, si la performance du succès ou de la performance d'accélération est prise en considération, CPS-Q_w est préférable ; si un taux

de réussite plus élevé et la stabilité de l'algorithme sont attendus, CPS-ρ_{rank} est suggéré. De plus, les performances de KA-CMA-ES utilisant PS et CPS sont comparées. Il est évident que CPS surpasse PS pour la plupart des problèmes de test en fonction de la performance d'accélération. Ainsi, le contrôle d'impact de modèle est suggéré dans KA-CMA-ES en utilisant la présélection (voir Figure 6.7).

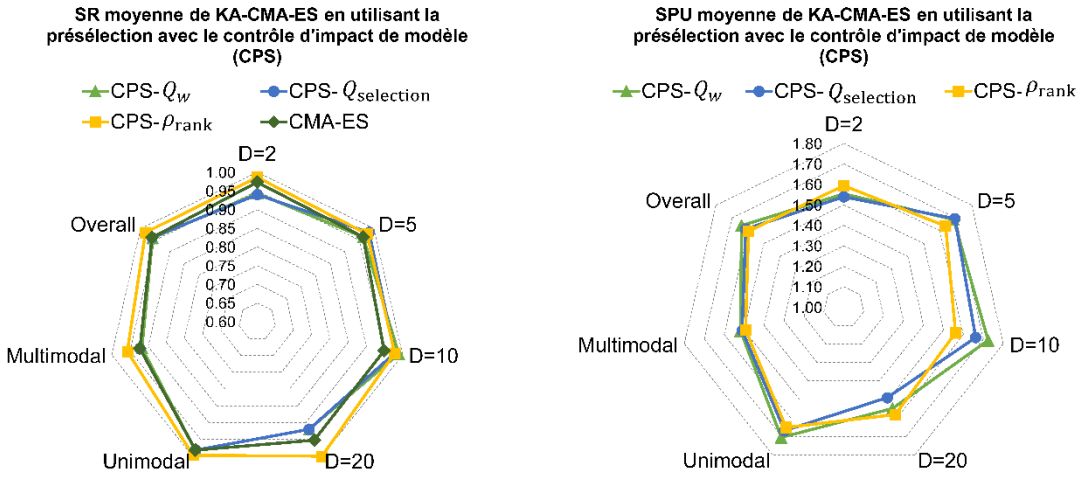

Figure 6.6 Taux moyen de réussite et taux de présélection supérieur à celui du contrôle de l'impact du modèle à l'aide de différentes mesures de la qualité du modèle.

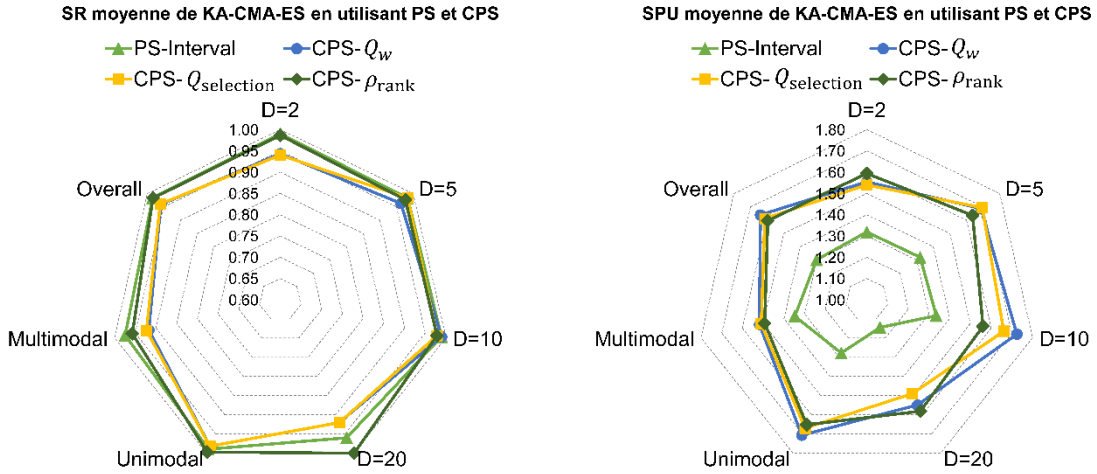

Figure 6.7 Taux moyen de réussite et accélération de la présélection sans et avec le contrôle de l'impact du modèle.

Pour les KA-CMA-ES utilisant un contrôle à base individuelle fixe (FIC), cinq paramètres (moyenne, écart-type, SLB, POI et IE) sont étudiés. Les résultats sont présentés à la Figure 6.8. Parmi les cinq métriques, ces métriques (SLB, POI et EI) qui équilibrent

l'exploitation du substitut et l'exploration de l'espace de recherche sont meilleures que les métriques qui ne concilient que l'exploitation ou l'exploration (moyenne et SD). On prend en considération SR et SP, FIC-EI présente des avantages par rapport aux autres paramètres.

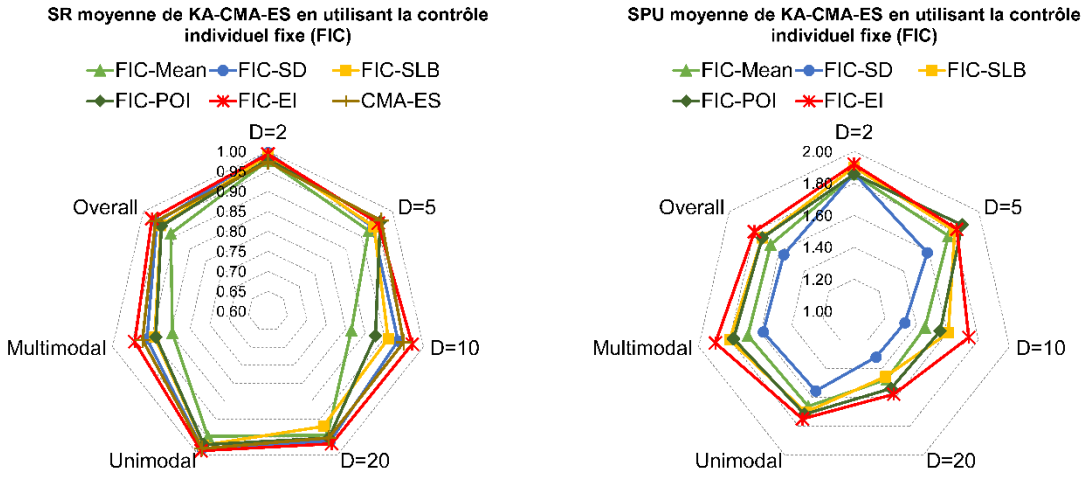

Figure 6.8 Taux moyen de réussite (SR) et performance d'accélération (SPU) de KA-CMA-ES en utilisant un contrôle individuel fixe avec différentes mesures.

Dans notre contrôle individuel mixte (MIC), deux paramètres possibles, la moyenne et la SLB, sont examinés. Les résultats expérimentaux (voir Figure 6.9) ont prouvéque le MIC-SLB a un taux de succès plus élevéque celui de MIC-Mean, en particulier sur des problèmes multimodaux. De plus, en général, MIC-SLB a une meilleure performance d'accélération moyenne que MIC-Mean. En tenant compte à la fois du taux de réussite et des performances d'accélération, on peut conclure que MIC-SLB surpasse MIC-Mean. À partir de la comparaison (voir Figure 6.10) de KA-CMA-ES en utilisant FIC et MIC, le FIC-EI a généralement le taux de réussite le plus élevé. D'autres algorithmes de contrôle individuels ont un taux de réussite moyen modérésupérieur à 85 %. Cependant, KA-CMA-ES utilisant la MIC surpasse de manière significative FIC en fonction de la performance d'accélération et de la réussite. En particulier, MIC-SLB a la moyenne la plus élevée SPU et le taux de réussite acceptable. Par conséquent, MIC-SLB est préférable parmi les algorithmes de KA-CMA-ES en utilisant un contrôle individuel.

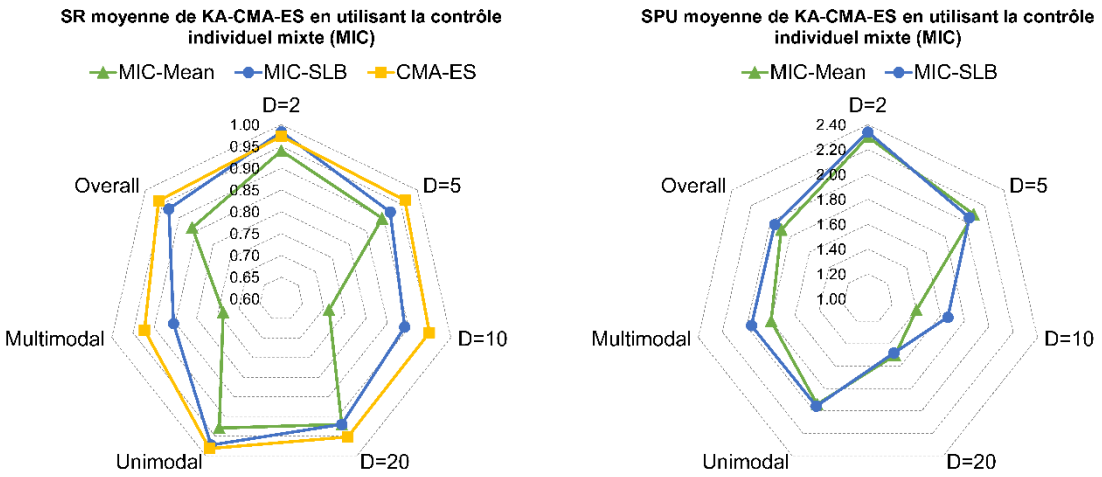

Figure 6.9 Taux moyen de réussite (SR) et performance d'accélération (SPU) de KA-CMA-ES en utilisant un contrôle mixte individuel (MIC).

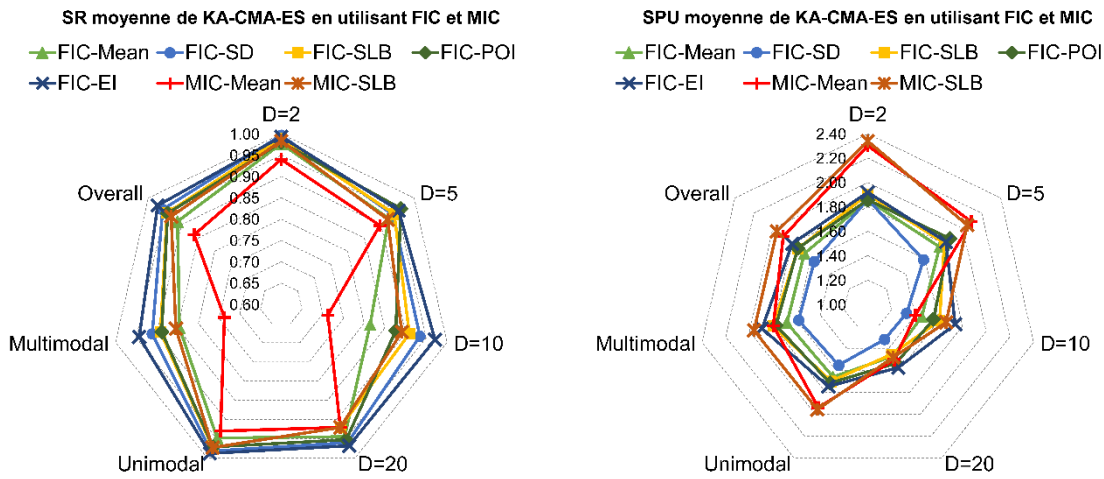

Figure 6.10 Moyenne SR et SPU de KA-CMA-ES en utilisant un contrôle individuel fixe et mixte (FIC et MIC).

Nous avons modifié la procédure de classement approximatif et l'avons ensuite intégrée au CMA-ES, appelé KA-CMA-ES en utilisant la procédure de classement approximative (ARP). Outre la métrique Mean, EI métrique suggérée précédemment est utilisée dans KA-CMA-ES à l'aide de l'ARP. Il a montré que (voir Figure 6.11) l'ARP-EI a une meilleure performance que l'ARP-Mean, selon le taux de réussite et le performance d'accélération.

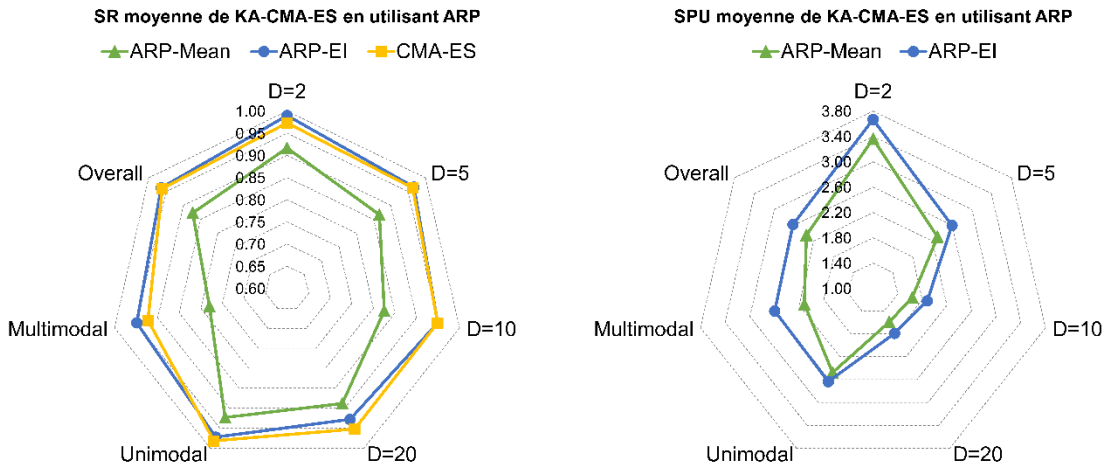

Figure 6.11 Taux moyen de réussite et performance d'accélération de KA-CMA-ES en utilisant ARP.

De plus, le KA-CMA-ES utilisant un contrôle basé sur la génération a été étudié, où l'on examine et compare le contrôle basé sur la génération fixe (FGC) et le contrôle de génération basé sur la génération adaptée (AGC) (voir Figure 6.12). Généralement, le KA-CMA-ES utilisant AGC est plus performant que celui utilisant FGC. Le contrôle de génération adaptatif utilisant ρ_{rank} ($\text{AGC-}\rho_{\text{rank}}$) comme qualité de modèle a le taux de réussite moyen le plus élevé sur tous les problèmes. Considérant les performances d'accélération, le contrôle de génération adaptatif en utilisant Q_w et $Q_{\text{selection}}$ ont de meilleures performances. Dans l'ensemble, $\text{AGC-}\rho_{\text{rank}}$ est préférable en fonction du taux de réussite et de la stabilité, et est préférable lorsque les performances d'accélération sont prises en considération.

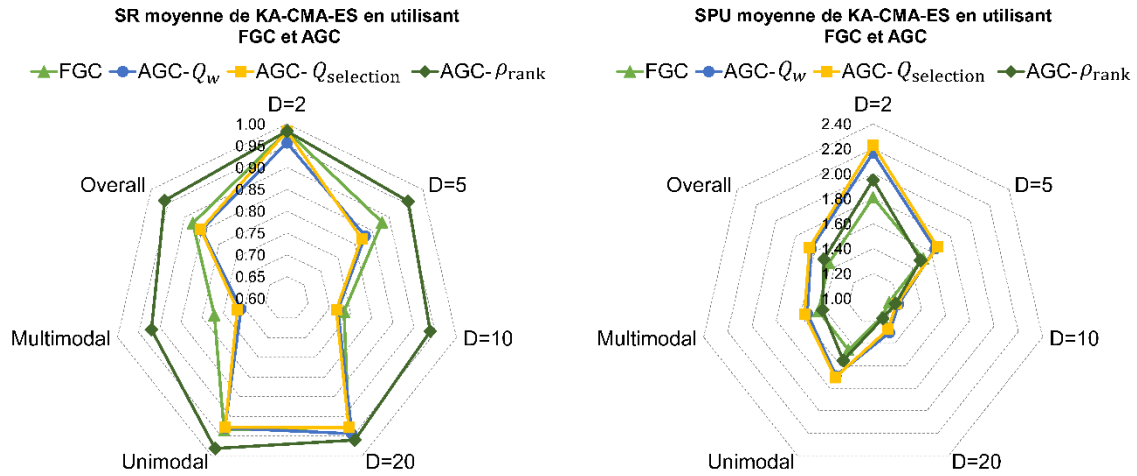

Figure 6.12 Taux moyen de réussite et accélération de KA-CMA-ES en utilisant FGC et AGC.

Enfin, tous les algorithmes étudiés de KA-CMA-ES sont analysés. Sur la base de la discussion précédente et les résultats, seuls les meilleurs algorithmes de chaque catégorie sont choisis comme les représentants et utilisés dans l'analyse et la comparaison ici. L'intervalle PS est pris comme représentant de la présélection sans contrôle de l'impact du modèle (PS). CPS-Q_w et CPS-ρ_{rank} sont considérés comme des algorithmes typiques de présélection avec le contrôle d'impact modèle (CPS). Le FIC-EI est utilisé pour représenter le contrôle individuel fixe (FIC). L'ARP-EI désigne le KA-CMA-ES en utilisant la procédure de classement approximatif (ARP). Pour le contrôle basé sur la génération, AGC-Q_w et AGC-ρ_{rank} sont considérés comme les représentants. A partir des résultats expérimentaux, il est évident que l'ARP-EI présente des performances d'accélération exceptionnelles parmi tous les algorithmes étudiés (voir Figure 6.14). Les valeurs SPU de PS-Intervalle, CPS-Q_w, CPS-ρ_{rank}, FIC-EI et MIC-SLB sur tous les problèmes de test sont supérieures à un (voir Figure 6.13). Ceci montre une amélioration stable de la performance du succès de ces algorithmes. Les stabilités de AGC-Q_w et AGC-ρ_{rank} sont inférieurs à d'autres. Il n'y a qu'un seul problème sur lequel l'ARP-EI se détériore plus que le CMA-ES. Par conséquent, on peut affirmer que l'ARP-EI est exceptionnelle parmi tous les algorithmes KA-CMA-ES étudiés, en fonction de la performance d'accélération ou de la performance de succès. Compte tenu à le taux de réussite et le performance de l'accélération, on peut conclure que l'ARP-EI est préférable parmi tous les algorithmes KA-CMA-ES étudiés dans cette thèse.

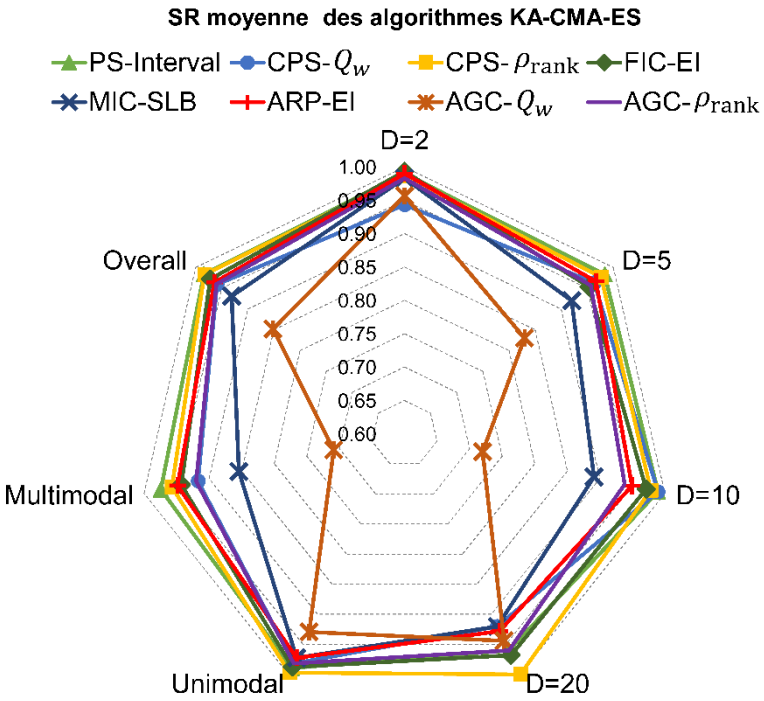

Figure 6.13 Comparaison du taux de réussite moyen (SR) des algorithmes KA-CMA-ES.

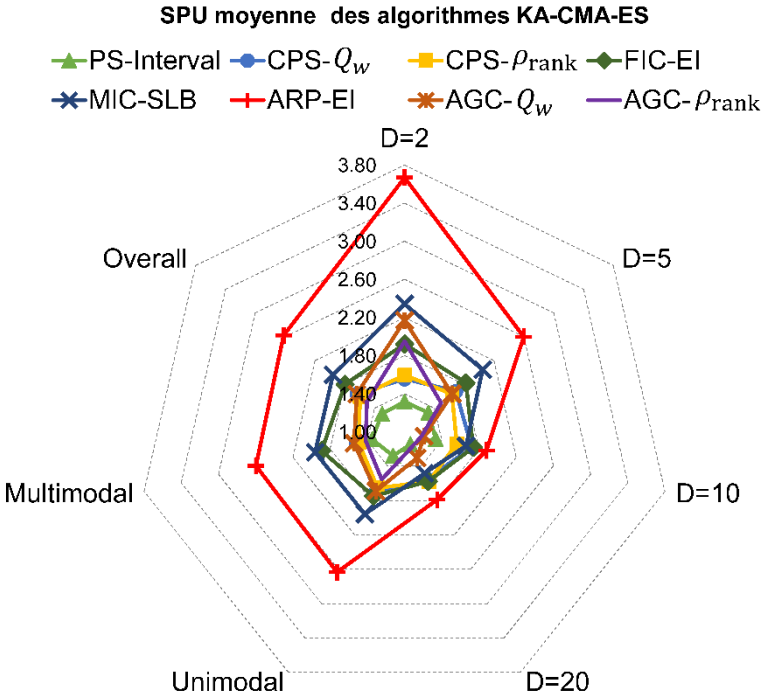

Figure 6.14 Comparaison de la performance d'accélération moyenne (SPU) des algorithmes KA-CMA-ES.

6.2.5 Cinquième Chapitre: Application dans l'Identification des Paramètres du Matériau

Ce chapitre présente les applications de l'algorithme KA-CMA-ES proposé dans l'identification des paramètres matériels qui est un problème d'optimisation coûteux. Un modèle d'endommagement élastique-plastique, dans lequel le comportement de durcissement de la matière est décrit par la loi de Swift et le dommage ductile est modélisé par le modèle de Lemaitre, est présenté et implémenté dans la simulation numérique par le sous-programme VUMAT dans ABAQUS. Ensuite, le KA-CMA-ES utilisant la méthode de classement approximatif avec métrique EI (ARP-EI) est appliqué en méthode inverse d'identification de paramètre de matériau.

6.2.5.1 Modèle d'Endommagement Elastique-Plastique

En général, un modèle constitutif élastique-plastique normal contient les composants de base suivants :

1) La loi élastique, qui donne la relation entre la contrainte et la contrainte élastique ;
2) Le critère de rendement, qui définit la limite du comportement plastique ;
3) La règle d'écoulement, qui décrit l'évolution de la déformation plastique ;
4) La loi de durcissement, qui caractérise le développement du critère de rendement ultérieur.

En conséquence, le modèle d'endommagement élastique-plastique contient une loi d'évolution des variables d'endommagement en plus des quatre composants ci-dessus.

Dans ce travail, le critère de rendement de Von Mises couramment utilisé, la loi d'écrouissage de Swift et la loi d'endommagement ductile de Lemaitre sont inclus dans le modèle élastique-plastique avec endommagement. Les équations constitutives du modèle élastique-plastique acev endommagement sont présentées au Tableau 6.1.

Tableau 6.1 Equations constitutives du modèle d'endommagement élastique-plastique

Décomposer de tenseur	$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p$
Loi élastique avec endommagement	$\boldsymbol{\sigma} = (1-D)\mathbf{E} : \boldsymbol{\varepsilon}^e$
La règle d'écoulement	$\Phi = q/(1-D) - \sigma_y$
loi de durcissemen (loi de Swift)	$\sigma_y\left(\bar{\varepsilon}^p\right) = A\left(\varepsilon_0 + \bar{\varepsilon}^p\right)^n$
Evolution de la déformation plastique	$\dot{\boldsymbol{\varepsilon}}^p = \dfrac{\dot{\gamma}}{(1-D)}\sqrt{\dfrac{3}{2}}\dfrac{\boldsymbol{s}}{\|\boldsymbol{s}\|}$
Évolution des endommagement s	$\dot{D} = \dfrac{\dot{\gamma}}{1-D}\left(\dfrac{-Y}{r}\right)^s \hat{H}\left(\bar{\varepsilon}^p - \bar{\varepsilon}_D^p\right)$
Conditions de chargement/déchargement	$\Phi \le 0,\ \dot{\gamma} \ge 0,\ \Phi\dot{\gamma} = 0$

6.2.5.2 Méthode Inverse d'Identification

Le but de la procédure d'identification des paramètres est de réduire le plus possible la distance entre les réponses simulées et expérimentales. En d'autres termes, le processus d'identification des paramètres consiste à trouver un ensemble de paramètres qui fait la différence minimale entre la réponse de simulation et les données d'expérience. Ainsi, la méthode inverse est un outil utile pour déterminer les paramètres du matériau. La méthode inverse d'identification du paramètre considère l'identification des paramètres comme un problème d'optimisation, qui consiste à trouver un ensemble de paramètres qui minimisent la différence entre les données expérimentales et les simulations numériques en norme. Le cadre de la méthode inverse d'identification du paramètre est illustré par la Figure 6.15.

6.2.5.3 Identification des Paramètres

Identification des Paramètres de Durcissement

L'identification inverse des paramètres de durcissement est effectuée indépendamment avec le CMA-ES standard et le KA-CMA-ES en utilisant l'ARP-EI. Dans le processus d'identification, des simulations FEM du test de traction sont effectuées à plusieurs reprises pour évaluer la fonction objectif. Les limites inférieure et supérieure des paramètres dans le processus d'identification et les paramètres de durcissement de contrainte identifiés sont données dans le Tableau 6.2. Les graphiques de convergence et l'historique itératif des paramètres sont illustrés à la Figure 6.16. Les courbes force-déplacement correspondant aux

paramètres de durcissement de contrainte identifiés et à la courbe expérimentale sont présentées à la Figure 6.17.

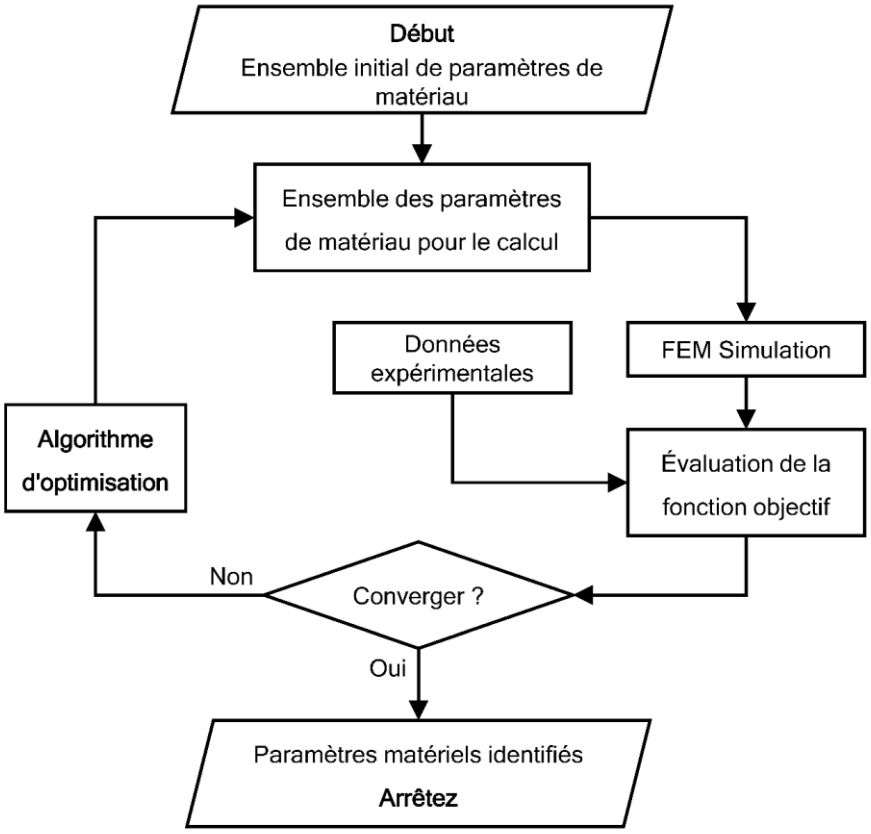

Figure 6.15 La procédure d'identification des paramètres

Tableau 6.2 Résultats et coûts de calcul de l'identification des paramètres de durcissement

Paramètre	A	ε_0	n	Nombre d'évaluations de la fonction objective
Limites inférieures	600	0	0.1	–
Limites supérieures	900	0.03	0.4	–
La CMA-ES	754.1402	0.0087	0.2264	533
KA-CMA-ES utilisant ARP-EI	754.2124	0.0087	0.2264	211

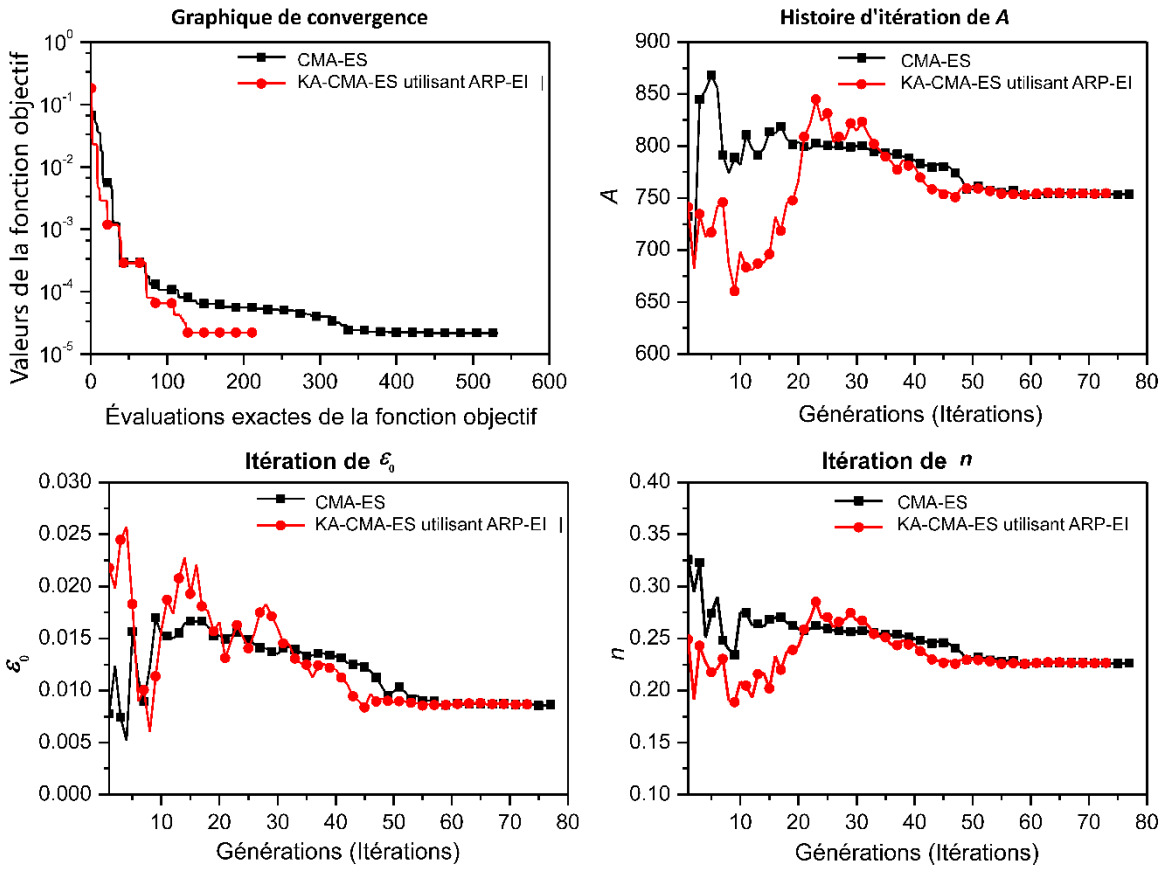

Figure 6.16 Graphes de convergence et d'itération de l'identification des paramètres de durcissement

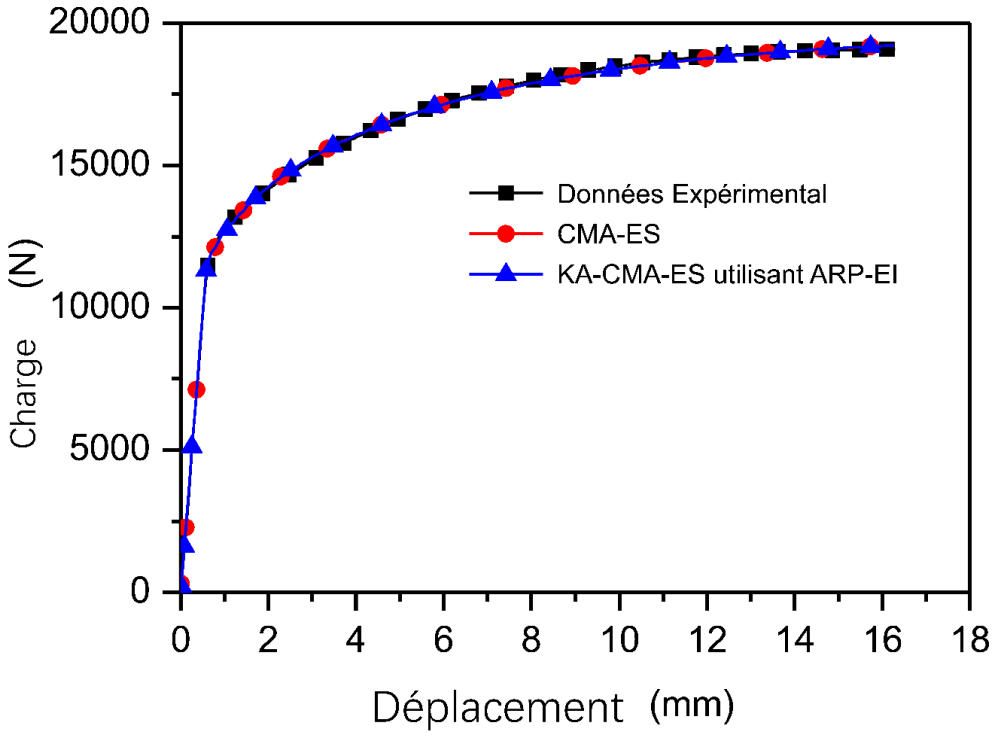

Figure 6.17 Courbes de déplacement de force (avant étranglement) correspondant à des paramètres de durcissement de contrainte identifiés et données expérimentales

Identification des Paramètres d'Endommagement

Après que les paramètres de durcissement de contrainte sont identifiés dans la sous-section précédente, nous nous intéressons à l'identification des paramètres d'endommagement ductiles $\bar{\varepsilon}_D^p$, r et D_C. Dans le modèle d'endommagement ductile, on utilise le critère de rendement de Von Mises et la loi de durcissement de Swift. Les paramètres identifiés dans la paragraphe précédente sont utilisés ici. Les limites inférieure et supérieure pour les paramètres dans le processus d'identification et les paramètres d'endommagement de contrainte identifiés sont données dans le Tableau 6.3.

Tableau 6.3 Résultats et coûts de calcul de l'identification des paramètres d'endommagement

Paramètre	$\bar{\varepsilon}_D^p$	r	D_C	Nombre d'évaluations de la fonction objective
Limites inférieures	0.1	2	0	–
Limites supérieures	0.18	10	0.8	–
La CMA-ES	0.1306	6.1575	0.0874	295
KA-CMA-ES utilisant ARP-EI	0.1303	6.1583	0.0898	167

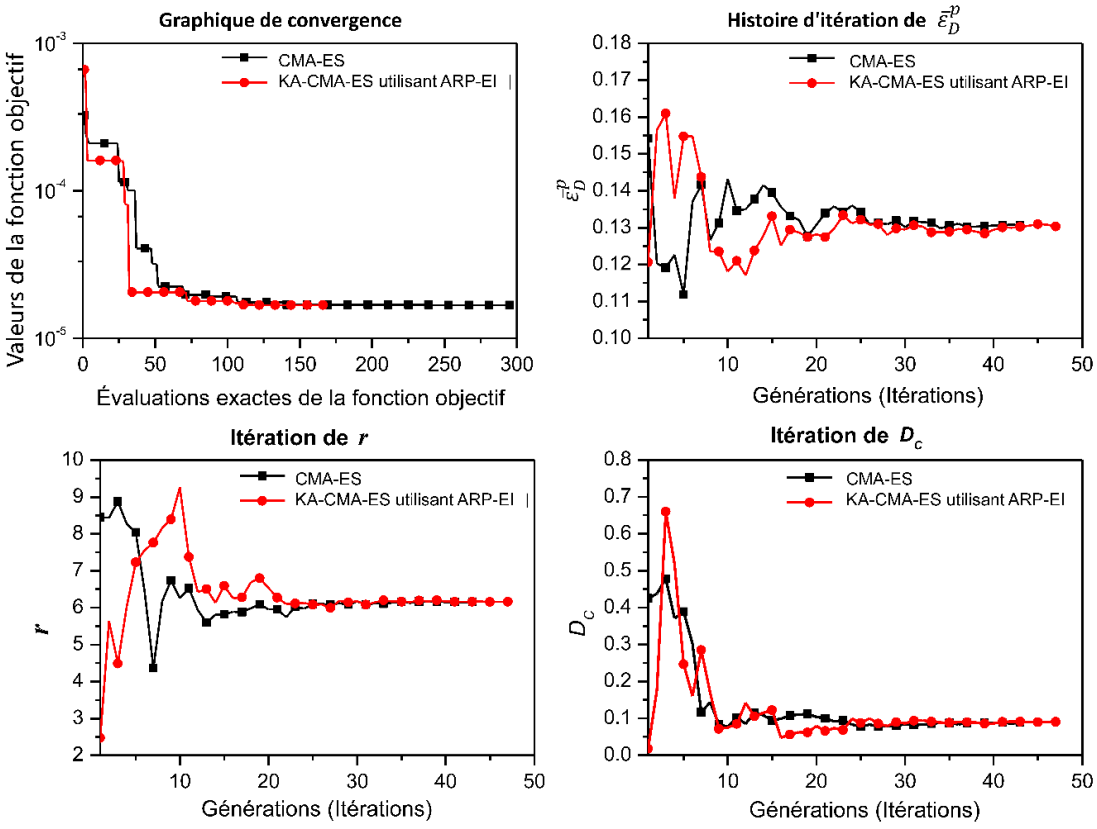

Figure 6.18 Graphes de convergence et d'itération de l'identification des paramètres d'endommagement

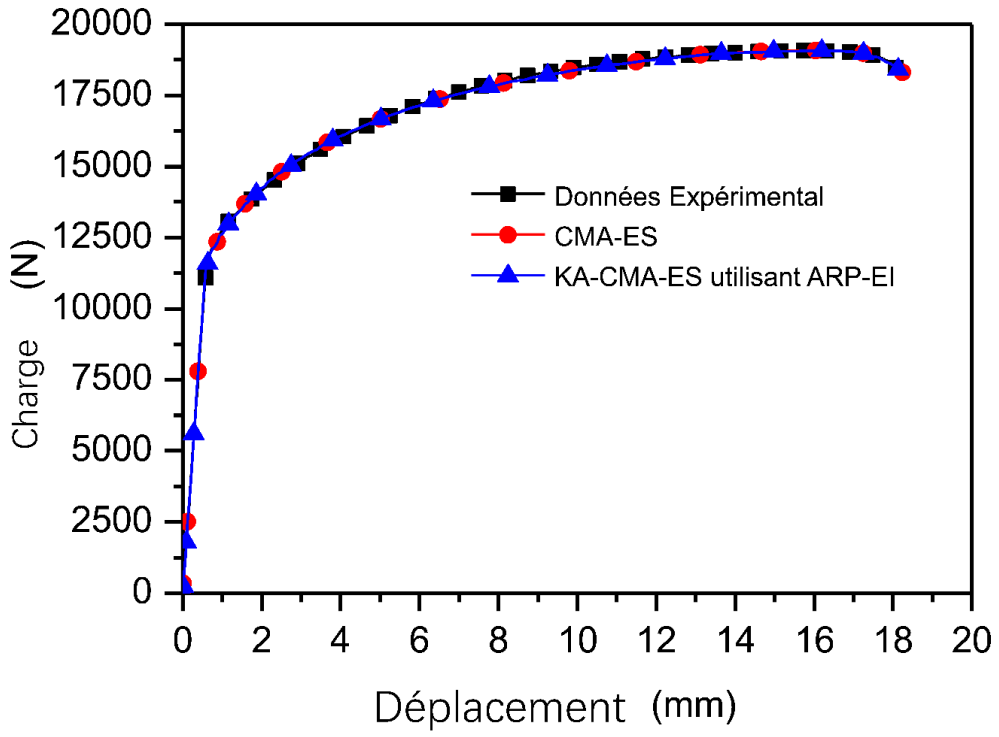

Figure 6.19 Courbes de force-déplacement correspondant aux paramètres d'endommagement identifiés et données expérimentales

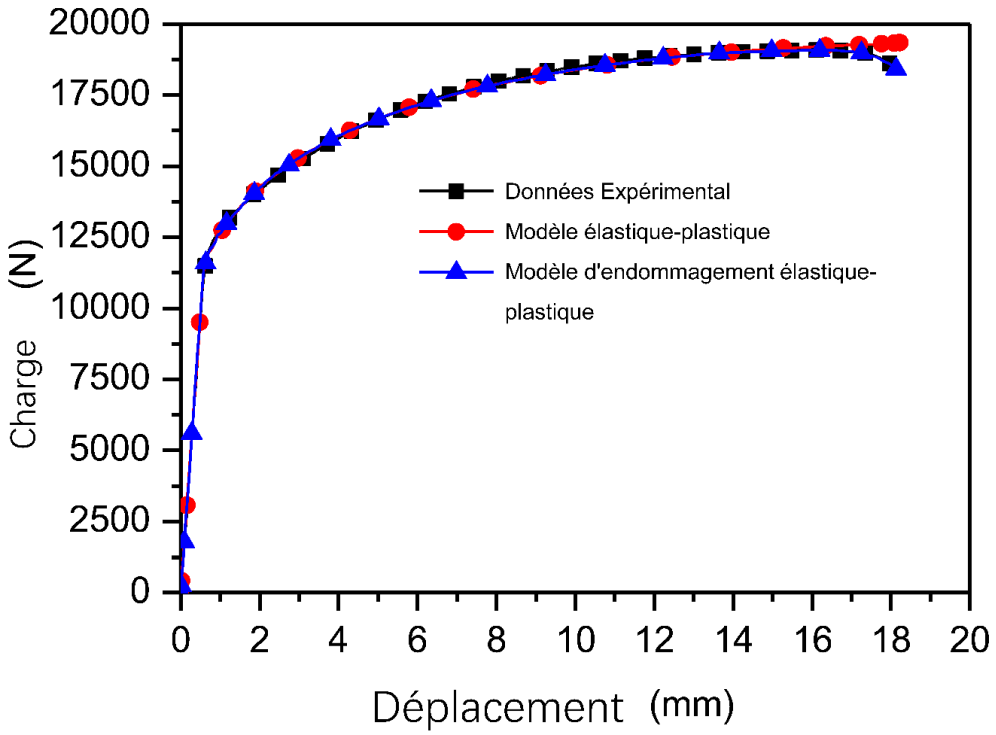

Figure 6.20 Comparaison des courbes force-déplacement des modèles élastique-plastique et élastique-plastique avec les paramètres identifiés

Les graphiques de convergence et l'historique itératif des paramètres d'endommagement sont illustrés à la Figure 6.18. Les courbes force-déplacement

correspondant aux paramètres d'endommagement élastique-plastique identifiés et à la courbe expérimentale sont présentées par la Figure 6.19. De plus, les résultats de la simulation du modèle élastique-plastique et du modèle élastique-dégradation plastique ainsi que les données expérimentales sont également représentées dans la Figure 6.20.

En conclusion, le modèle d'endommagement élastique-plastique présenté est approprié pour la modélisation du comportement du matériau en tenant compte du durcissement et des effets de détérioration, et le KA-CMA-ES proposé à l'aide d'ARP-EI est fiable et efficace en méthode inverse d'identification des paramètres.

6.3 Conclusions et perspectives

Nous avons développé de nouveaux algorithmes qui combinent la stratégie d'évolution d'adaptation de matrice de covariance (CMA-ES) et le modèle de substitution de krigeage afin de réduire le nombre d'évaluations de la fonction de condition physique (fonction objectif) et ainsi d'améliorer l'efficacité de résoudre des problèmes coûteux.

Conclusions

Les résultats de cette étude ont démontré que la méthode de l'intervalle de confiance proposée pour la sélection des ensembles d'apprentissage semble supérieure aux méthodes couramment utilisées «Récemment» et «kNN» pour la sélection d'ensembles d'apprentissage. Ainsi, la méthode de l'intervalle de confiance est suggérée pour la sélection de l'ensemble d'apprentissage dans le calcul évolutif assisté par substitution. A partir des résultats expérimentaux, on peut affirmer que l'ARP-EI est remarquable parmi tous les algorithmes KA-CMA-ES étudiés, en fonction de la performance d'accélération ou de réussite.

L'application de KA-CMA-ES en utilisant l'ARP-EI en méthode inverse d'identification des paramètres pour le modèle d'endommagement élastique-plastique montre également que le KA-CMA-ES améliore significativement l'efficacité de CMA-ES et réduit ainsi considérablement le coût de calcul lors de l'identification des paramètres matériels.

Perspectives

Les algorithmes KA-CMA-ES développés ont permis d'améliorer l'efficacité de la norme CMA-ES. Cependant, il y a encore des questions ouvertes. Ici, nous aimerions mentionner les problèmes de recherche future, à partir de notre point de vue.

Tout d'abord, il n'existe aucune conclusion claire quant à savoir quel modèle de substitution est meilleur dans l'optimisation évolutionnaire assistée par modèles substitut. Par conséquent, d'autres modèles substituts doivent être étudiés dans l'optimisation évolutionnaire assistée par un substitut.

En ce qui concerne le KA-CMA-ES dédoublé dans ce travail, l'étude expérimentale et l'application sur des problèmes dimensionnels élevés (plus de 20 dimensions) pourraient être réalisées dans les travaux futurs.

Pour les problèmes de dimension élevée, le coût de calcul de la formation du modèle de substitution augmenterait apparemment. Cela peut devenir un inconvénient des algorithmes d'évolution assistée par substitution. Par conséquent, l'optimisation d'évolution assistée par substitution pour le problème dimensionnel élevé est un sujet digne d'étude.

Conclusions and Perspectives

In this thesis, we have comprehensively explored the Kriging-Assisted Covariance Matrix Adaptation Evolution Strategy (KA-CMA-ES) for expensive optimization problems. Furthermore, engineering application of the developed new algorithm is performed in material parameter identification. This final chapter gives a summary of what we have learned from our works and points out the most promising direction for future research.

Conclusions

We have developed new algorithms, which combine Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and Kriging surrogate model, to reduce the number of fitness function (objective function) evaluations and thus to improve the efficiency of solving expensive problems. In Kriging-Assisted CMA-ES (KA-CMA-ES), Kriging models are repeatedly learned based on previously evaluated data points and then used to predict the fitness of new individuals instead of original fitness function evaluations. In this way, the number of expensive fitness function evaluations is significantly decreased and thus the computational cost is cut down. Different methods of model exploitation (the different ways of incorporating Kriging model into the standard CMA-ES), which brings about different KA-CMA-ES algorithms, have been investigated in our work. The KA-CMA-ES algorithms investigated in this thesis can be divided into four groups: KA-CMA-ES using pre-selection, KA-CMA-ES using individual-based control, KA-CMA-ES using approximate ranking procedure, and KA-CMA-Es using generation-based control. In the experimental studies of KA-CMA-ES, different aspects of KA-CMA-ES algorithms have been examined by running the experiments on 40 benchmarking problems (including 12 test function with different dimensions).

In the study of KA-CMA-ES using pre-selection without model impact control (PS), different training set selection methods, including ‘Recently’ (the Recently Evaluated

Points), ‘kNN’ (the k-Nearest Neighbor Points to distribution mean based on Mahalanobis distance), and ‘Interval’ (the proposed Confidence Interval method), have been investigated. The results of this investigation demonstrated that the proposed confidence interval method for training set selection apparently superior to the commonly used ‘Recently’ and ‘kNN’ methods for training set selection. KA-CMA-ES using PS with ‘Interval’ has higher success rate and better success performance than that using ‘Recently’ and ‘kNN’. Thus, the confidence interval method is suggested for training set selection in surrogate-assisted evolutionary computation.

In the study of KA-CMA-ES using pre-selection with model impact control (CPS), three model quality measures, Q_w , $Q_{\text{selection}}$ and ρ_{rank} , are used and compared. The results show that, if the success performance or speedup performance is considered, CPS-Q_w is preferable; if higher success rate and the stability of algorithm are expected, CPS-ρ_{rank} is suggested. Additionally, the performance of KA-CMA-ES using PS and CPS are compared. It is apparent that pre-selection with model impact control outperforms pre-seleciton without model impact control for most of the test problems according to the speedup performance. Thus, model impact control is suggested in KA-CMA-ES using pre-selection.

For KA-CMA-ES using fixed individual-based control (FIC), five metrics (Mean, SD, SLB, POI, and EI) are studied. Among the five metric, these metric (SLB, POI and EI) which balances the exploitation of the surrogate and exploration of search space performs better than metrics that only consider exploitation or exploration (Mean, and SD). Take both SR and SP into consideration, FIC-EI has advantages over other metrics.

In our proposed mixed individual-based control (MIC), two possible metrics, Mean and SLB, are examined. The experimental results proved that MIC-SLB has higher success rate than that of MIC-Mean, especially on multimodal problems. Furthermore, generally, MIC-SLB has better average speedup performance than MIC-Mean. Taking both the success rate and speedup performance into consideration, it can be concluded that MIC-SLB outperforms MIC-Mean. From the comparison of KA-CMA-ES using FIC and MIC, the FIC-EI generally has the highest success rate. Other individual-based control algorithms have moderate

average success rate larger than 0.85. However, KA-CMA-ES using MIC significantly outperform FIC according to the speedup performance and success performance. Particularly, MIC-SLB has the highest average SPU and acceptable success rate. Therefore, MIC-SLB is preferable among algorithms of KA-CMA-ES using individual-based control.

We have modified the approximate ranking procedure and then embedded it into the CMA-ES, which is referred to as KA-CMA-ES using approximate ranking procedure (ARP). Besides the Mean metric, the previously suggested metric EI is used in KA-CMA-ES using ARP. It has been concluded that ARP-EI outperforms ARP-Mean, considering both the success rate and success performance.

Moreover, the KA-CMA-ES using generation-based control has been investigated, where the fixed generation-based control (FGC) and the proposed adaptive generation-based control (AGC) are examined and compared. Generally, the KA-CMA-ES using AGC performs better than that using FGC. The adaptive generation control using ρ_{rank} (AGC-ρ_{rank}) as model quality has the highest average success rate on all the problems. Considering the speedup performance, adaptive generation control using Q_w and $Q_{\text{selection}}$ have better performance. Overall, AGC-ρ_{rank} is preferable according to success rate and stability, and AGC-Q_w is preferable when speedup performance is considered.

Finally, all the investigated algorithms of KA-CMA-ES are analyzed. Based on previously discussion and results, only the best algorithms of each category of algorithms are chosen as the representatives and used in analysis and comparison here. The PS-Interval is taken as the representative of pre-selection without model impact control (PS). The CPS-Q_w and CPS-ρ_{rank} are considered as typical algorithms of pre-selection with model impact control (CPS). The FIC-EI is used to represent the fixed individual-based control (FIC). The ARP-EI stands for the KA-CMA-ES using approximate ranking procedure (ARP). For generation-based control, AGC-Q_w and AGC-ρ_{rank} are regarded as the representatives. From the experimental results, it is apparent that ARP-EI has outstanding speedup performance among all the investigated algorithms. The SPU values of PS-Interval, CPS-Q_w , CPS-ρ_{rank} , FIC-EI and MIC-SLB on all the test problems are larger than one. This shows a

stable improvement in success performance of these algorithms. The stabilities of AGC-Q_w and AGC-ρ_{rank} are lower than others. There is only one problem on which ARP-EI performs worsen than CMA-ES. Therefore, it can be stated that the ARP-EI is outstanding among all the investigated KA-CMA-ES algorithms, according to the speedup performance or success performance. Considering both the average SR and SPU, it can be concluded that ARP-EI is preferable among all the investigate KA-CMA-ES algorithms in this work.

The application of KA-CMA-ES using ARP-EI in inverse method of parameter identification for the elastic-plastic damage model also shows that the KA-CMA-ES improve the efficiency of CMA-ES significantly and thus greatly reduce the computational cost of material parameter identification.

Perspectives

The developed KA-CMA-ES algorithms have apparently improve the efficiency of the standard CMA-ES. However, there are still some open questions. Here we would like to mention several issues of future research, from our view of points.

Firstly, with respect to the surrogate models used in surrogate-assisted evolutionary computation, there is still no clear conclusion on which model is better than others. Thus, more surrogate models need to be investigated.

Regards to the KA-CMA-ES developed in this work, the experimental study and application on high dimensional problems (larger than 20 dimension) could be carried out in future works.

For high dimensional problems, the computational cost of surrogate model training would increase apparently. This may become a disadvantage of surrogate-assisted evolution algorithms. Therefore, surrogate-assisted evolutionary optimization for high dimensional problem is a topic worthy of studying.

Bibliography

1. Arora RK (2015) Optimization: Algorithms and Applications. CRC Press
2. Tenne Y, Goh C-K (2010) Computational Intelligence in Expensive Optimization Problems. Springer. doi: 10.1007/978-3-642-10701-6
3. Bäck T, Foussette C, Krause P (2013) Contemporary Evolution Strategies. Natural Computing Series. doi: 10.1007/978-3-642-40137-4
4. Koziel S, Yang X-S (2011) Computational Optimization, Methods and Algorithms. doi: 10.1007/978-3-642-20859-1
5. Chong EKP, Żak SH (2001) An introduction to optimization. John Wiley & Sons
6. Diest K (2013) Numerical Methods for Metamaterial Design. doi: 10.1007/978-94-007-6664-8
7. Nocedal J, Wright SJ Numerical Optimization. doi: 10.1007/978-0-387-40065-5
8. Papadrakakis M, Pantazopoulos G (1993) Survey of quasi-Newton methods with reduced storage. International Journal for Numerical Methods in Engineering 36:1573–1596. doi: 10.1002/nme.1620360910
9. Nelder JA, Mead R (1965) A Simplex Method for Function Minimization. The Computer Journal 7:308–313. doi: 10.1093/comjnl/7.4.308
10. Yang X-S (2010) Engineering Optimization. doi: 10.1002/9780470640425
11. Rios LM, Sahinidis N V. (2013) Derivative-free optimization: A review of algorithms and comparison of software implementations. Journal of Global Optimization 56:1247–1293. doi: 10.1007/s10898-012-9951-y
12. Conn a. R, Scheinberg K, Toint PL (1997) Recent progress in unconstrained nonlinear optimization without derivatives. Mathematical Programming 79:397–414. doi: 10.1007/BF02614326
13. Torczon V (1997) On the convergence of pattern search algorithms. SIAM Journal on Optimization 7:1–25. doi: 10.1137/S1052623493250780

14. Audet C, Dennis JE (2006) Mesh Adaptive Direct Search Algorithms for Constrained Optimization. SIAM Journal on Optimization 17:188–217. doi: 10.1137/040603371

15. Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications 79:157–181. doi: 10.1007/BF00941892

16. Huyer W, Neumaier A (1999) Global Optimization by Multilevel Coordinate Search. Journal of Global Optimization 14:331–355. doi: 10.1023/A:1008382309369

17. Jin Y, Sendhoff B (2002) Fitness Approximation in Evolutionary Computation: A Survey. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO. pp 1105–1113

18. Jin Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9:3–12. doi: 10.1007/s00500-003-0328-5

19. Jin Y (2011) Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1:61–70. doi: 10.1016/j.swevo.2011.05.001

20. Shi L, Rasheed K (2010) A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms. In: Computational intelligence in expensive optimization. pp 3–28

21. Jin Y, Olhofer M, Sendhoff B (2000) On evolutionary optimization with approximate fitness functions. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO. Morgan Kaufmann, pp 786–793

22. Dennis J, Dennis JE, Torczon V, Torczon V (1995) Managing Approximation Models in Optimization. In: Multidisciplinary Design Optimization: State-of-the-Art. pp 330–347

23. Yang BS, Yeun YS, Ruy WS (2002) Managing approximation models in multiobjective optimization. Structural and Multidisciplinary Optimization 24:141–156. doi: 10.1007/s00158-002-0224-0

24. Gräning L, Jin Y, Sendhoff B (2007) Individual-based Management of Meta-models for Evolutionary Optimization with Application to Three-Dimensional Blade Optimization. In: Studies in Computational Intelligence. pp 225–250

25. Ulmer H, Streichert F, Zell A (2004) Optimization by Gaussian Processes assisted Evolution Strategies. In: Selected Papers of the International Conference on Operations Research (OR 2003). pp 435–442

26. Rasheed K, Ni X, Vattam S (2005) Comparison of methods for developing dynamic reduced models for design optimization. Soft Computing 9:29–37. doi: 10.1007/s00500-003-0331-x

27. Rasheed K, Hirsh H (2000) Informed operators: Speeding up Genetic-algorithm-based design Optimization using Reduced Models. In: Genetic and Evolutionary Computation Conference. Morgan Kaufmann, pp 628–635

28. Ratle A (1998) Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In: Ppsn 1998, Lncs. pp 87–96

29. Kramer O (2016) Machine Learning for Evolution Strategies. doi: 10.1007/978-3-319-33383-0

30. Kramer O (2014) A Brief Introduction to Continuous Evolutionary Optimization. doi: 10.1007/978-3-319-03422-5

31. Hansen N, Arnold D V., Auger A (2015) Evolution Strategies. In: Springer Handbook of Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 871–898

32. Man KF, Tang KS, Kwong S (1999) Genetic Algorithms. Springer. doi: 10.1007/978-1-4471-0577-0

33. Vanneschi L, Systems E (2012) Handbook of Natural Computing. Handbook of Natural Computing. doi: 10.1007/978-3-540-92910-9

34. Beyer H-G (2001) The Theory of Evolution Strategies. Natural Computing Series. doi: 10.1007/978-3-662-04378-3

35. Rencher AC (2002) Methods of multivariate analysis, 2nd ed. John Wiley & Sons

36. Rechenberg I (1973) Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog

37. Ostermeier A, Gawelczyk A, Hansen N (1994) Step-size adaptation based on non-local use of selection information. In: Parallel Problem Solving from Nature — PPSN III. Springer Berlin Heidelberg, pp 282–291

38. Loshchilov I (2013) Surrogate-Assisted Evolutionary Algorithms. Inria Saclay

39. Hansen N, Ostermeier A (2001) Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9:159–195. doi: 10.1162/106365601750190398

40. Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation. IEEE, pp 312–317

41. Hansen N (2016) The CMA Evolution Strategy: A Tutorial.

42. Hansen N (2006) The CMA Evolution Strategy: A Comparing Review. In: Towards a New Evolutionary Computation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 75–102

43. Fang K-TK, Li RZ, Sudjianto A, et al (2006) Design And Modeling for Computer Experiments. CRC Press

44. Box GEP, Draper NR (1987) Empirical Model-Building and Response Surfaces. Wiley Series in Probability and Mathematical Statistics. doi: 10.1037/028110

45. Giunta AA, Wojtkiewicz S, Eldred MS (2003) Overview of modern design of experiments methods for computational simulations. Aiaa 649:6–9. doi: 10.2514/6.2003-649

46. Sacks J, Welch WJ, Mitchell JSB, Henry PW (1989) Design and Experiments of Computer Experiments. Statistical Science 4:409–423. doi: 10.2307/2245858

47. Dean A, Morris M, Stufken J, Bingham D (2015) Handbook of Design and Analysis of Experiments. CRC Press

48. Santner TJ, Williams BJ, Notz WI (2003) The Design and Analysis of Computer Experiments. Series in Statistics. doi: 10.1007/978-1-4757-3799-8

49. Huang C, Radi B, Hami A El (2016) Uncertainty analysis of deep drawing using surrogate model based probabilistic method. The International Journal of Advanced Manufacturing Technology 86:3229–3240. doi: 10.1007/s00170-016-8436-4

50. Messac A (2015) Optimization in Practice with MATLAB®: For Engineering Students and Professionals. Cambridge University Press

51. SEEGER M, Rasmussen CE, Williams KI (2004) Gaussian processes for machine learning. International journal of neural systems. doi: 10.1142/S0129065704001899

52. Kleijnen JPC (2009) Kriging metamodeling in simulation: A review. European Journal of Operational Research 192:707–716. doi: 10.1016/j.ejor.2007.10.013

53. Jones DR, Schonlau M, Welch WJ (1998) Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13:455–492. doi: 10.1023/a:1008306431147

54. Forrester AIJJ, Keane AJ (2009) Recent advances in surrogate-based optimization. Progress in Aerospace Sciences 45:50–79. doi: 10.1016/j.paerosci.2008.11.001

55. Martin JD, Simpson TW (2003) A Study on the Use of Kriging Models to Approximate Deterministic Computer Models. In: Volume 2: 29th Design Automation Conference, Parts A and B. ASME, pp 567–576

56. Wu Y, Wang H, Zhang B, Du K-L (2012) Using Radial Basis Function Networks for Function Approximation and Classification. ISRN Applied Mathematics 2012:1–34. doi: 10.5402/2012/324194

57. Forrester AIJ, Sóbester A, Keane AJ (2008) Engineering Design via Surrogate Modelling： A Practical Guide. John Wiley & Sons

58. Engelbrecht AP (2007) Computational intelligence : an introduction, Second Edi. John Wiley & Sons

59. Haykin S (2008) Neural Networks and Learning Machines, Third Edit. Pearson Upper Saddle River, NJ, USA

60. Santana-Quintero L V, Montaño AA, Coello CAC (2010) A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In: Computational Intelligence in Expensive Optimization Problems. pp 29–59

61. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Statistics and Computing 14:199–222. doi: 10.1023/B:STCO.0000035301.49549.88

62. Vapnik VN (2000) The Nature of Statistical Learning Theory. doi: 10.1007/978-1-4757-3264-1

63. Boyd SP, Vandenberghe L (2004) Convex optimization. Cambridge University Press

64. Ryberg A-B, Bäckryd RD, Nilsson L (2015) A metamodel-based multidisciplinary design optimization process for automotive structures. Engineering with Computers 31:711–728. doi: 10.1007/s00366-014-0381-y

65. Wang GG, Shan S (2007) Review of Metamodeling Techniques in Support of Engineering Design Optimization. Journal of Mechanical Design 129:370. doi: 10.1115/1.2429697

66. Lin Y (2004) An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design. Georgia Institute of Technology

67. Bäck T, Schwefel H-P (1993) An Overview of Evolutionary Algorithms for Parameter Optimization. Evolutionary Computation 1:1–23. doi: 10.1162/evco.1993.1.1.1

68. Du K-L, Swamy MNS (2016) Evolutionary Strategies. In: Search and Optimization by Metaheuristics. Springer International Publishing, Cham, pp 83–91

69. Papadrakakis M, Lagaros ND, Tsompanakis Y (1998) Structural optimization using evolution strategies and neural networks. Computer Methods in Applied Mechanics and Engineering 156:309–333. doi: 10.1016/S0045-7825(97)00215-6

70. Yaochu Jin, Olhofer M, Sendhoff B (2001) Managing approximate models in evolutionary aerodynamic design optimization. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). IEEE, pp 592–599

71. Bajer L, Pitra Z, Holeňa M (2015) Investigation of Gaussian Processes and Random Forests as Surrogate Models for Evolutionary Black-Box Optimization. In: Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15. ACM Press, New York, New York, USA, pp 1351–1352

72. Ulmer H, Streichert F, Zell A (2003) Evolution strategies assisted by Gaussian processes with improved preselection criterion. In: The 2003 Congress on Evolutionary Computation, 2003. CEC '03. IEEE, pp 692–699

73. Emmerich M, Giotis A, Özdemir M, et al (2002) Metamodel-Assisted Evolution Strategies. In: Guervós JJM, Adamidis P, Beyer H-G, et al (eds) In Parallel Problem Solving from Nature VII. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 361–370

74. Ulmer H, Streichert F, Zell A (2005) Model Assisted Evolution Strategies. In: Knowledge Incorporation in Evolutionary Computation. pp 333–355

75. Runarsson TP (2004) Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. In: Parallel Problem Solving from Nature - PPSN VIII. pp 401–410

76. Kern S, Hansen N, Koumoutsakos P (2006) Local Meta-models for Optimization Using Evolution Strategies. In: Runarsson TP, Beyer H-G, Burke E, et al (eds) Parallel Problem Solving from Nature - PPSN IX. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 939–948

77. Bouzarkouna Z, Auger A, Ding DY (2010) Investigating the Local-Meta-Model CMA-ES for Large Population Sizes. In: Di Chio C, Cagnoni S, Cotta C, et al (eds) Applications of Evolutionary Computation: EvoApplicatons 2010. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 402–411

78. Loshchilov I, Schoenauer M, Sebag M (2010) Comparison-Based Optimizers Need Comparison-Based Surrogates. In: Parallel Problem Solving from Nature, PPSN XI. Springer Berlin Heidelberg, pp 364–373

79. Runarsson TP (2006) Ordinal Regression in Evolutionary Computation. In: Proceedings of Parallel Problem Solving from Nature-PPSN IX. pp 1048–1057

80. Loshchilov I, Schoenauer M, Sebag M (2012) Self-adaptive Surrogate-assisted Covariance Matrix Adaptation Evolution Strategy. Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12) 321–328. doi: 10.1145/2330163.2330210

81. Lu J, Li B, Jin Y (2013) An evolution strategy assisted by an ensemble of local Gaussian process models. In: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13. ACM Press, New York, New York, USA, p 447

82. Jin Y, Michael H, Sendhoff B (2003) Quality measures for approximate models in evolutionary computation. In: Proceedings of the Bird of a Feather Workshop, Genetic and Evolutionary Computation Conference. AAAI, Chicago, pp 170–173

83. Spearman C (1904) The Proof and Measurement of Association between Two Things. The American journal of psychology 15:72–101. doi: 10.1037/h0065390

84. Ulmer H, Streichert F, Zell a. (2004) Evolution strategies with controlled model assistance. Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat No04TH8753) 2:1569–1576. doi: 10.1109/CEC.2004.1331083

85. Yaochu Jin, Olhofer M, Sendhoff B (2002) A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation 6:481–494. doi: 10.1109/TEVC.2002.800884

86. Kern S, Hansen N, Koumoutsakos P (2004) Fast quadratic local meta-models for evolutionary optimization of anguilliform swimmers. EUROGEN 2007

87. Loeppky JL, Sacks J, Welch WJ (2009) Choosing the Sample Size of a Computer Experiment: A Practical Guide. Technometrics 51:366–376. doi: 10.1198/TECH.2009.08040

88. Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation 4:150–194. doi: 10.1504/IJMMNO.2013.055204

89. Suganthan PN, Hansen N, Liang JJ, et al (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report 2005005:2005.

90. Chen Q, Liu B, Zhang Q, et al (2014) Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization.

91. Murakami S (2012) Continuum Damage Mechanics. Journal of Chemical Information and Modeling. doi: 10.1007/978-94-007-2666-6

92. De Souza Neto EA, Peric D, Owen DRJ (2011) Computational methods for plasticity: theory and applications. John Wiley & Sons

93. Teixeira PMC (2010) Ductile Damage Prediction in Sheet Metal Forming and Experimental Validation. Universidade do Porto

94. Lemaitre J, Desmorat R (2005) Engineering damage mechanics: ductile, creep, fatigue and brittle failures. Springer Science & Business Media

95. Lemaitre J (1983) A three-dimensional ductile damage model applied to deep-drawing forming limits. Mechanical Behavior of Materials--IV, 2:1059–1065.

96. Stein E, De Borst R, Hughes TJR (2004) Encyclopedia of computational mechanics. John Wiley & Sons

97. Radi B, Ayadi M, Cherouat A, et al (2011) Identification of the Characteristics of Hydroformed Structures Using optimization Methods. Key Engineering Materials 473:723–730.

98. El Hami A, Bouchaib R (2013) Uncertainty and Optimization in Structural Mechanics. John Wiley & Sons

99. Ledesma A, Gens A, Alonso EE (1996) Estimation of parameters in geotechnical backanalysis—I. Maximum likelihood approach. Computers and Geotechnics 18:1–27.

100. Yang Z, Elgamal A (2003) Application of unconstrained optimization and sensitivity analysis to calibration of a soil constitutive model. International journal for numerical and analytical methods in geomechanics 27:1277–1297.

101. Kachanov L (2013) Introduction to continuum damage mechanics. Springer Science & Business Media

102. Lemaitre J (1984) How to use damage mechanics. Nuclear engineering and design 80:233–245.

103. Bouchard P-O, Bourgeon L, Fayolle S, Mocellin K (2011) An enhanced Lemaitre model formulation for materials processing damage computation. International Journal of Material Forming 4:299–315.

104. Runesson K, Steinmann P, Ekh M, Menzel A (2006) Constitutive modeling of engineering materials--theory and computation. Volume I General Concepts and Inelasticity

105. Cekerevac C, Girardin S, Klubertanz G, Laloui L (2006) Calibration of an elasto-plastic constitutive model by a constrained optimisation procedure. Computers and Geotechnics 33:432–443.

106. Radi B, Cherouat A, El Hami A, Mahfoudh A (2010) Optimization technics to identify the characteristics of a hydroformed structure. International Journal for Simulation and Multidisciplinary Design Optimization 4:39–47.

107. Zhou J-M, Qi L-H, Chen G-D (2006) New inverse method for identification of constitutive parameters. Transactions of Nonferrous Metals Society of China 16:148–152.

108. Roux É, Bouchard P-O (2010) Ductile Damage Material Parameter Identification: Numerical Investigation. In: Tenth International Conference on Computational Structures Technology. Paper--135