Surrogate-Assisted Evolutionary Algorithms For Computationally Expensive Optimisation Problems

By

Xunzhao Yu

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
February 2023

ABSTRACT

Surrogate-assisted evolutionary algorithms (SAEAs) are designed to solve black-box optimisation problems that are computationally expensive. These optimisation problems could be multi-objective or/and constrained. In industry, many real-world optimisation problems, such as engine calibration problems, allow only a few evaluations due to their high cost and therefore limited budgets. To save these valuable budgets, this thesis focuses on developing effective and efficient SAEAs to solve computationally expensive optimisation problems. This thesis makes three major contributions: First, to achieve better optimisation results within limited evaluations, it proposes a domination-based ordinal regression surrogate for SAEAs. This ordinal regression surrogate approximates the ordinal landscape of multiple objectives. Two surrogate management strategies are also proposed to cooperate with the ordinal regression surrogate. The resulting SAEAs outperform the state-of-the-art algorithms on a set of multi-objective optimisation problems. Second, it develops a bilevel SAEA to handle constrained optimisation problems. In the algorithm, decision variables are divided into either upper-level or lower-level variables according to their impacts on solution feasibility. The lower-level optimisation focuses on lower-level variables to make candidate solutions feasible, while the upper-level optimisation adjusts upper-level variables to optimise objective(s). The algorithm finds more feasible solutions and better optimisation results in an engine calibration problem when compared with the algorithms used in the engine industry and some state-of-the-art algorithms. Third, for the sake of saving more evaluations, it proposes an experience-based SAEA framework to learn experience from related tasks and then use the learned experience in a new optimisation task. The experience is learned through a novel meta-learning technique and they represent the domain-specific features among related tasks. Assisted by the learned experience, accurate surrogates can be learned with very few evaluated samples in an efficient way. Experimental studies have demonstrated the effectiveness of experience learning and that of using experience in an existing SAEA. Competitive optimisation results are achieved while fewer evaluations are used than before, which saves valuable evaluation budgets considerably.

In memory of my grandfather, Xiu-Song Yu

(Chinese Calendar, 5th April 1945 to 7th January 2021)

ACKNOWLEDGMENTS

Firstly, I would like to express my deepest gratitude to my grandfather in heaven, who raised me up and taught me how to read and write. Thanks to his attention and great efforts on my education. Without his support, I would not have the opportunity to pursue my undergraduate and postgraduate education in universities. He was the most cherished and esteemed person in my life. I love and admire him forever not only for his meticulous care, but also for his admirable character and qualities. Due to the flight cancellations caused by COVID-19, I failed to keep him company while he was fighting with cancers. This is the biggest regret in my life.

Secondly, I want to thank my supervisor Prof. Xin Yao. Thanks to his patient supervision and valuable advice on conducting research work. His knowledge and expertise helped me a lot during my PhD study.

Thanks very much also to my co-supervisors, Prof. Per Kristian Lehre and Prof. Joshua Knowles. I obtained many valuable suggestions about the research methodologies and received many encouragements from the discussions with them.

Thanks to Dr. Yan Wang, the technical expert from the Ford motor company who cooperated with me in my research projects. He provided me with many insightful comments from the perspective of industry, making me realize the differences in perspectives between academia and industry when resolving real-world problems. Thanks to Ford USA for funding my PhD programme.

I would also like to thank some excellent colleagues and friends who helped me during my PhD study. Thanks to Prof. Chao Qian for his discussions on the research issues I met and for his impressive suggestions on my research career. Thanks to Dr. Xiaodong Jia and

Dr. Weiqi Chen for helping me accustomed to the daily life in U.K. at the early stage of my PhD study. Thanks to Dr. Miqing Li for his discussions and useful suggestions on my research projects.

Finally, I wish to thank my grandmother, Sai-Yu Lin, for her effort to raise me up. Thanks to my parents, my uncle, my aunts, and my elder brother for their consistent support and encouragement. They are willing to hear me from my homeland. Special thanks to my lovely wife, Yuan Fang, for her care and support during my PhD study.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Overall

$\hat{\mu}, \hat{\sigma}$ Estimations of μ, σ.

$\hat{y}, \hat{s}^2$ Estimations of mean and covariance produced by a GP model.

$\mathcal{N}(\mu, \sigma^2)$ A normal distribution with mean μ and variance σ^2.

$\Phi(\cdot), \phi(\cdot)$ Cumulative distribution function, probability density function.

$\phi(\mathbf{w}, \mathbf{b})$ A neural network with weights $\mathbf{w}$ and biases $\mathbf{b}$.

$\boldsymbol{\theta}, \mathbf{p}$ Coefficient, exponent parameter vectors of a GP model.

$\mathbf{R}, \mathbf{r}$ Correlation matrix, correlation vector of a GP model.

$\mathbf{X}$ Domain of optimisation problem f.

$\mathbf{x}$ Vector of decision variables.

$\mathbf{x}^i$ The i^{th} vector of decision variables.

$\mathbf{y}$ Vector of dependent variables.

d Number of decision variables / Dimensionality of vector $\mathbf{x}$.

$f(\cdot)$ Optimisation problem/ Objective functions.

$f_i(\cdot)$ The i^{th} objective function of optimisation problem f.

FE Number of used fitness evaluations.

FE_{max} Number of fitness evaluation budget.

$g(\cdot)$ Inequality constraints of the optimisation problem.

$h(\cdot)$ Surrogate model.

$k(\cdot)$ Equality constraints of the optimisation problem.

$L(\cdot)$ Loss function.

m Number of objectives.

n Number of samples.

N_o Number of ordinal levels in the ordinal surrogate.

n_o Minimum of ordinal levels in the ordinal surrogate.

N_{init} Size of the initial archive S_A.

S_A Archive of evaluated samples / solutions.

x_j^i The j^{th} element of vector $\mathbf{x}^i$.

Chapter3

λ Factor of smoothness in AOREA.

$\mathbf{b}_u, \mathbf{b}_l$ Upper and lower bounds of S_{AF}.

C_{shape} Shape criterion in the ordinal surrogate.

c_{shape} A coefficient for shaping reference points in the ordinal surrogate.

$f_0(\cdot)$ Objective functions that have been centralised by $\mathbf{b}_l$.

$g_\lambda(\cdot)$ Smooth function in AOREA.

n_{rep} Number of solutions to be reproduced in the adaptive evolutionary search in AOREA.

r_i The i^{th} region in S_R.

RV Reference vectors for dividing objective space into several regions (constant).

s Optimisation state in AOREA.

s_m Minimum optimisation state in AOREA.

S_R Set of sub-regions in the objective space.

S_V Set of relation values.

S_{AF_0} Attainment front that has been centralised by $\mathbf{b}_l$.

S_{AF} Attainment front.

S_{PF} Pareto front.

S_{RP} Set of reference points.

Chapter4

ϵ Ratio of infeasible solutions used in ϵ-selection.

imp Quantified impacts of decision variables.

$\mathbf{P}_{X_l}$ Probability of allocating a variable into X_l.

$\mathbf{x}_l$ Vector of lower-level variables.

$\mathbf{x}_u$ Vector of upper-level variables.

$\mathbf{x}_u^i$ The i^{th} vector of upper-level decision variables.

E Elite population.

$f_l(\cdot)$ Objective function of the lower-level optimisation.

$f_u(\cdot)$ Objective function of the upper-level optimisation.

$h_l(\cdot)$ Lower-level surrogate model.

$h_u(\cdot)$ Upper-level surrogate model.

L Lower-level population.

N_e Size of the elite population E.

N_l Size of the lower-level population L.

N_u Size of the upper-level population U.

S_f Feasible solutions in archive S_A.

t_{max} Maximum iterations of lower-level optimisation.

U Upper-level population.

X_l Set of (the indexes of) lower-level variables.

X_u Set of (the indexes of) upper-level variables.

Chapter5

α, β Learning rates of neural networks.

$\Delta\boldsymbol{\theta}^*, \Delta\mathbf{p}^*$ Task-specific increments.

$\boldsymbol{\gamma}$ Vector of all parameters in a deep kernel.

$\boldsymbol{\theta}^*, \mathbf{p}^*, \boldsymbol{\gamma}^*$ Parameters that are trained on T_*.

$\boldsymbol{\theta}^e, \mathbf{p}^e, \boldsymbol{\gamma}^e$ Parameters used to represent experience.

$\mathbf{z}$ The last k variables in $\mathbf{x}$, where $k = d - m + 1$.

$|D_m|$ Size of dataset D_m.

A Amplitude of sinusoid functions.

B Number of related tasks in a batch (Batch size).

b Phase of sinusoid functions.

D_i Dataset collected from a related task T_i.

D_m Subsets sampled from D_i for meta-learning.

D_{mi} The i^{th} subset D_m, $m = 1, \ldots, N_m$.

e Mean square error value.

$k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma})$ Kernel k with parameters $\boldsymbol{\gamma}$ and inputs $\mathbf{x}^i, \mathbf{x}^j$.

N Number of related tasks.

N_m Number of subsets D_m for meta-learning.

S, Q Support set and query set.

S_*, Q_* Support set and query set collected from T_*.

T_* Target optimisation task.

T_i A related optimisation task.

U Number of update iterations in the meta-learning procedure.

w Frequency of sinusoid functions.

List of Abbreviations

ACO ..... Ant Colony Optimisation

AF ..... Attainment front

ANN ..... Artificial Neural Network

BSFC ..... Brake Special Fuel Consumption

CHT ..... Constraint Handling Technique

DACE ..... Design and Analysis of Computer Experiments

DE ..... Differential Evolution

DoE ..... Design of Experiment

DTLZ ..... Deb Thiele Laumanns Zitzler

EA ..... Evolutionary Algorithm

ECOP ..... Expensive Constrained Optimisation Problem

EI ..... Expected Improvement

EMOP ..... Expensive Multi-objective Optimisation Problem

EMaOP ..... Expensive Many-objective Optimisation Problem

EMTO ..... Evolutionary Multi-Tasking Optimisation

EP ..... Evolutionary Programming

ES ..... Evolutionary Strategy

FNN ..... Feedforward Neural Network

GA ..... Genetic Algorithm

GP ..... Gaussian Process

HV ..... Hypervolume

IGD+ ..... Inverted Generational Distance Plus

KNN ..... K-Nearest Neighbour

LHS ..... Latin Hypervolume Sampling

MOEA ..... Multi-Objective Evolutionary Algorithm

MSE ..... Mean Squared Error

NMSE ..... Normalised Mean Squared Error

NN ..... Neural Network

PCA ..... Principle Component Analysis

PF ..... Pareto front

PoI ..... Probability of Improvement

PSO ..... Particle Swarm Optimisation

RBF ..... Radial Basis Function

SAEA ..... Surrogate-Assisted Evolutionary Algorithm

SBX ..... Simulated Binary Crossover

SVM ..... Support Vector Machine

Chapter 1

Introduction

1.1 Background

In real-world applications, numerous optimisation problems are sophisticated such that their mathematical expressions are not available. These optimisation problems are called black-box optimisation problems as we cannot assume they have properties such as linearity, continuity, convexity, or even differentiability. Hence, it is difficult to solve these optimisation problems with traditional gradient-based optimisation methods. In contrast, evolutionary algorithms (EAs) are a family of powerful heuristic optimisation algorithms that are capable of solving black-box optimisation problems. Inspired by biological phenomena, EAs are usually implemented by either emulating the process of evolution or behaving like animals in a herd. They are able to approximate the global optima of black-box problems without requiring gradient information. Due to these advantages, EAs have been widely applied to a variety of real-world applications successfully, such as engine calibration [116], traffic signal control [86], production scheduling [55], water resource engineering [39], and medical resource management [37].

In EAs, solutions are evaluated to obtain their performance on the optimisation problems to be solved. To reduce the risk of getting stuck in local optima, EAs usually evaluate a

large number of solutions during the evolutionary search. Moreover, some features of black-box optimisation problems can lower the optimisation efficiency of EAs. For example, when solving constrained black-box optimisation problems with EAs, quite a lot of evaluations can be wasted on infeasible solutions. Because EAs often do not know whether a solution is feasible or not until the solution is evaluated. Additionally, those problems with many decision variables usually have a very large decision space. Due to the curse of dimensionality, EAs need to evaluate more solutions to sufficiently search over a large decision space. Therefore, most EAs require a large number of evaluations.

However, the cost of evaluations is not trivial in some real-world applications. Take the engine calibration problem as an example: Evaluating engine performance on engine testing facilities is costly in finance and also time-consuming [116]. Considering the limited financial budget and short calibration cycles, only a few performance evaluations will be allowed during the whole calibration process. As a result, engine calibration problems and other evaluation-costly optimisation problems can be formulated as expensive black-box optimisation problems.

The limitation of evaluation budget prevents most EAs from solving expensive optimisation problems directly. An intuitive idea to handle expensive optimisation problems is to employ computational efficient surrogate models in EAs [42, 54]. During the evolutionary optimisation, the solutions that have been evaluated on expensive functions can be maintained as an archive. In some EAs, such an archive serves as the training dataset for computational efficient surrogates. By approximating the expensive problem or expensive fitness functions [42], these surrogates are able to produce approximate objective values for given solutions with a much lower cost than expensive optimisation problems. The tributary of EAs assisted by these surrogates is referred to as surrogate-assisted evolutionary algorithm (SAEA). In practice, for an iteration of evolutionary optimisation in SAEAs, the offspring solutions that originally were evaluated on expensive functions will be firstly evaluated on computational efficient surrogates. Based on the surrogate evaluation results, only a few offspring solutions

will be selected by SAEAs and then re-evaluated on expensive functions. Here, the environmental selection criterion used to determine which offspring solutions are good enough to be selected is known as the infill sampling criterion. Consequently, many expensive evaluations are replaced by computational efficient surrogate evaluations, which saves valuable evaluations in expensive optimisation problems. The strategy of using computational efficient surrogates to enhance the efficiency of using expensive function evaluations is known as surrogate management in conventional optimisation or evolution control in evolutionary computation [45, 43].

Although SAEAs are capable of handling expensive black-box optimisation problems, it should be noted that they still have some challenges in expensive optimisation problems. Most challenges are caused by other characteristics of optimisation problems. For example, some expensive optimisation problems have more than one conflicting objectives, which are known as expensive multi-objective (two or three objectives) or many-objective (more than three objectives) optimisation problems (EMOPs or EMaOPs). In EMOPs or EMaOPs, the evaluation of every objective function is computationally costly and it is essential to find proper approximations for all objectives. Besides, the diversity of non-dominated solutions in the objective space is very important to the optimisation of multiple objectives. During the optimisation, the balance between convergence and diversity also deserves concern. Some expensive optimisation problems have constraints in the objective space. These problems are expensive constrained optimisation problems (ECOPs). The constraints in ECOPs can be equality, inequality, linear, and nonlinear. Also, these constraints can be soft or hard constraints. The SAEAs designed for ECOPs have to consider the surrogate modelling for constraints and how to handle constraints with surrogates. In addition, some expensive optimisation problems suffer from the curse of dimensionality. These problems are large scale problems since they have many decision variables. One challenge to large scale problems is the efficiency of optimisation. The time cost of using some surrogates increases exponentially as the number of decision variables increases. Additionally, the performance of EAs can

also be affected adversely. Therefore, efforts should be made to deal with these challenges. Moreover, the decision space of some expensive optimisation problems is not continuous or even numerical. These problems are combinatorial optimisation problems and the domain of their decision variables is a set of finite elements. Therefore, conventional SAEAs designed for continuous optimisation cannot be applied to combinatorial optimisation problems directly.

Apart from the capability of handling diverse expensive optimisation problems, optimisation performance is also an important topic of SAEA research. To enhance the effectiveness and efficiency of SAEAs, various surrogates and surrogate management strategies have been developed and used in SAEAs. Moreover, some SAEAs even extract useful experience from other optimisation problems and use such experience to enhance the optimisation efficiency of new problems, which are referred to as experience-based SAEAs.

This thesis aims to solve some aforesaid challenges. Section 1.2 points out the scope of expensive optimisation problems considered in this thesis. In Section 1.3, the concrete research questions to be addressed in each chapter are clarified. Section 1.4 lists the major contributions made in this thesis. The thesis organisation and content summaries for each chapter are given in Section 1.5 In Section 1.6, the published papers and submitted papers included in this thesis are listed.

1.2 Thesis Scope

Despite the diverse challenges and problems discussed above, this thesis focuses on only the following three problems. The first problem is the expensive (unconstrained) multi-objective optimisation problem (EMOP). EMOP is one of the most common expensive optimisation problems. The second problem is the expensive constrained optimisation problem (ECOP). ECOP is also a kind of common expensive optimisation problem and it is widely existing in real-world applications. Different from our first problem, ECOP is more complex since optimisation algorithms need to search for optimal solutions on the basis of satisfying all

constraints. To concentrate on constraint handling techniques, the constrained optimisation problem to be solved has only one objective but multiple inequality constraints. The last problem is experience-based optimisation. Although our previous two problems are dealing with the most common and representative optimisation scenarios in expensive optimisation problems, they still only focus on the optimisation problems to be solved (denoted as target optimisation problems). All surrogate modelling and optimisation operations are started from scratch. To further improve the optimisation efficiency and performance of existing SAEAs, in our last problem, we take other optimisation problems that are related to the target optimisation problems into consideration. We aim to make SAEAs capable of gaining experience from related optimisation problems and then solve target optimisation problems efficiently via making good use of gained experience. Note that experience-based optimisation and transfer optimisation have been widely used in many practical applications, including dynamic optimisation and expensive optimisation problems [27]. In this thesis, we consider only the experience-based optimisation designed for expensive optimisation problems. It should be noted that the three problems considered in this thesis are neither large scale optimisation problems nor combinatorial optimisation problems. Besides, other characteristics of optimisation problems that have not been discussed above are not considered when solving these three problems.

1.3 Research Questions

Based on the thesis scope mentioned above, we clarify the concrete research questions to be addressed in this thesis as follows. The motivations and explanations of these research questions are also described.

1.3.1 Solving Expensive Multi-Objective Optimisation with Evolutionary Algorithms Assisted by Ordinal Surrogates

In the SAEAs that are designed to solve EMOPs, surrogates are expected to predict the ordinal relation between two given solutions. These surrogates help SAEAs to select either estimated optimal solutions for expensive fitness evaluations, or solutions for surrogate improvement. In most SAEAs, such a demand for surrogates is implemented by employing regression-based surrogates to approximate expensive objective functions. In practice, some SAEAs convert multiple objectives into a single objective through a weight vector, then a regression-based surrogate is used to approximate the aggregate function [47]. For two given solutions, their ordinal relation is accessible if the SAEA compares their surrogate evaluation results directly. Another way to use regression-based surrogates is to use multiple surrogates to approximate every objective function separately [120, 8]. When comparing two solutions, surrogates are used to estimate the fitnesses of two solutions on every objective function. A dominance-based comparison is then conducted to obtain the ordinal relation between two solutions.

Recently, many researchers realised that using regression-based surrogates is not necessary for the SAEAs designed for EMOPs. The demand for ordinal relation prediction can be satisfied by classification-based surrogates. Some SAEAs use classification-based surrogates to predict whether a solution is good or not based on the dominance-based relation between this solution and some reference solutions [68]. Another way to use classification-based surrogates is to learn the pairwise relation between solutions [117, 31]. Compared with regression-based SAEAs, although classification-based SAEAs take advantage of learning solution relations directly, their drawbacks are obvious: The learning of pairwise relations indicates an exponential increase in the complexity of surrogate training and prediction. For relations between reference points and solutions, the solutions classified into the same category are not comparable.

To take advantage of both regression-based and classification-based surrogates, we aim to design a new surrogate to approximate the ordinal relation between solutions directly but also to overcome the shortcomings of classification-based surrogates. Furthermore, assisted by such a new surrogate, we aim to develop new SAEAs to solve EMOPs in a more efficient way than existing SAEAs. Clearly, this thesis firstly focuses on the following research questions:

- How to construct an efficient surrogate to approximate the ordinal landscape of multi-objective optimisation problems?
- How to balance convergence and diversity when using our ordinal surrogate in SAEAs to solve EMOPs?

For the first research question, the time complexity of surrogate training should not be exponential. Besides, for the sake of environmental selection, it is preferable to make the results of surrogate evaluation comparable for a population of offspring solutions. For the second research question, proper surrogate management strategies should be developed to consider both convergence and diversity. In this way, our ordinal surrogate can be fully exploited and the optimisation performance of our SAEAs can be enhanced.

1.3.2 Surrogate-Assisted Bilevel Evolutionary Algorithm for Expensive Constrained Optimisation

For some real-world applications, the distribution of local optima implies that some decision variables are playing more important roles than other variables [97]. Therefore, the feasibility of solutions could be mainly affected by only a subset of decision variables, which motivates us to develop an efficient bilevel framework to handle constraints for ECOPs. The bilevel optimisation framework divides the optimisation process into an upper-level optimisation and a lower-level optimisation. The upper-level optimisation adjusts only upper-level variables and

the lower-level optimisation optimises only lower-level variables. Such a bilevel architecture will allocate more optimisation resources to the lower-level optimisation, because only the optimal solutions in the lower-level optimisation can be treated as the candidate solutions in the upper-level optimisation. Therefore, if we can identify which decision variables have significant impacts on solution feasibility and then classify them as the lower-level variables, then more optimisation efforts would be made toward these variables and the efficiency of handling constraints can be enhanced.

Besides, considering the problem to be solved having multiple constraints, it is desirable to use our ordinal surrogate to approximate these constraints. Therefore, we have the following research questions:

- How to identify which decision variables have significant impacts on solution feasibility and thus divide them into lower-level variables?
- How to apply our ordinal surrogate to approximate multiple constraints?
- In the framework of using bilevel architecture to solve ECOPs, how to use surrogate evaluations to generate optimal solutions for expensive fitness evaluations?

For the first research question, it is necessary to find a way to analyze and quantify the impact of decision variables on solution feasibility. For the second research question, our ordinal surrogate cannot be applied to approximate multiple constraints directly since the feasible solutions are incomparable in the constraint space. Hence, some modifications should be made. For the last research question, an effective surrogate management strategy should be developed to save expensive evaluations. Particularly, the EAs used to search for candidate solutions should be able to explore potentially feasible regions such that the evolutionary search will not be trapped in local optimums.

1.3.3 Experience-Based Surrogate-Assisted Evolutionary Algorithms for Expensive Optimisation

Learning quickly is a characteristic of human intelligence, which can be attributed to the capability of learning and using experience. Skilled humans are able to solve a new task quickly as they can gain useful experience from the related tasks they have seen before. These experiences are used to enhance the efficiency of working on a new task. In comparison, novices need more time to solve a new task due to their lack of experience. From the perspective of solving expensive optimisation problems, we hope our SAEAs behave like skilled humans instead of novices. In this way, a new optimisation problem can be solved with fewer expensive fitness evaluations, which saves more fitness evaluations and makes it possible to solve some extremely expensive optimisation problems. Therefore, we are motivated to mimic the behaviour of skilled humans and want to enable SAEAs with the capability of learning experience. The SAEAs that are capable of using experience are referred to as experience-based SAEAs.

In real-world applications, many optimisation problems are related since they are dealing with the optimisation problems in the same domain. These optimisation problems usually share some domain-specific features which are beneficial to the optimisation of other related optimisation tasks. The last optimisation problem to be solved in this thesis assumes that all related tasks are small and they cannot provide existing knowledge or experience on domain-specific features. As a result, the following research questions should be considered:

- How to represent experience from related expensive optimisation tasks and adapt them to a new expensive optimisation task?
- How to use and update experience during the surrogate-assisted evolutionary optimisation?

For the first research question, it should be noted that each related task can provide only

a small set of samples. Hence, useful experience may not be learned from a single related task. The key point in this question is to extract the domain-specific features over plenty of different related tasks. The second research question concerns the compatibility of SAEAs and the experience learning technique. A suitable strategy should be developed to make full use of experience in SAEAs.

1.4 Thesis Contributions

This thesis makes the following major contributions by addressing the research questions discussed in Section 1.3.

- A novel ordinal-regression-based surrogate designed for EMOPs. A hybrid surrogate management strategy and an adaptive surrogate management strategy are developed to use the ordinal-regression-based surrogate, resulting in two efficient and effective SAEAs for EMOPs. Detailed contributions can be found in the end of Chapter 3.

- The framework of a novel SAEA to solve ECOPs with a bilevel architecture. Additionally, a sensitivity analysis method is developed to quantify the impact of decision variables on solution feasibility. The ordinal-regression-based surrogate is also adapted to approximate the ordinal landscape of constraints. Detailed contributions can be found in the end of Chapter 4.

- A novel meta-learning method to learn experience from related expensive tasks. An experience-based SAEA framework to combine regression-based SAEAs with our experience learning method. Detailed contributions can be found in the end of Chapter 5.

1.5 Thesis Outline

The remaining content of the thesis is organised as follows.

Chapter 2 firstly formulates the computational expensive optimisation problems to be solved in this thesis. Then, it provides some preliminaries to SAEAs. After that, it reviews existing research work related to this thesis, including SAEAs designed to solve expensive multi-objective or expensive constrained optimisation problems, and experience-based SAEAs that attempt to enhance the optimisation efficiency by using the experience gained from related optimisation problems. Finally, it ends with an introduction to the surrogate modelling methods used in this thesis, including the Kriging model and neural network.

Chapter 3 firstly introduces an ordinal-regression-based surrogate to approximate the dominance-based ordinal landscape for EMOPs. After that, two surrogate management strategies are proposed to cooperate with the ordinal surrogate and solve EMOPs. The first management strategy is a hybrid and static strategy to consider both convergence and diversity. The second strategy is an adaptive strategy which is more flexible and powerful than the first one. Comparison experiments are conducted between state-of-the-art SAEAs on DTLZ benchmark test problems are detailed in this chapter.

Chapter 4 proposes an efficient bilevel SAEA to handle a real-world ECOP: the engine calibration problem. A sensitivity analysis method is introduced to quantify the impact of decision variables on solution feasibility. Then decision variables are divided into either upper-level or lower-level variables. A bilevel architecture is designed to optimise upper-level and lower-level variables in nested order. In addition, the ordinal surrogate described in Chapter 3 is adapted to approximate multiple constraints. Experimental results are conducted to compare our SAEA with two optimisation approaches used in the engine industry and other state-of-the-art SAEAs proposed for ECOPs. Detailed results are reported.

Chapter 5 presents a new experience-based SAEA framework to solve extremely ex-

pensive optimisation problems. This chapter does not focus on any specific optimisation scenario but concentrates on learning experience from related tasks and using experience in SAEAs. Details about experience learning approach are given and the method to use experience in SAEAs is explained. Empirical experiments are displayed to show the effectiveness of the experience learning approach as well as that of the experience-based SAEA framework.

Finally, Chapter 6 concludes the work of the thesis and future research directions are suggested.

1.6 Papers Resulting from the Thesis

The published or submitted papers resulting from the thesis are listed as follows.

Referred or Submitted Journal Papers

- X. Yu, L. Zhu, Y. Wang, D. Filev, and X. Yao, "Internal Combustion Engine Calibration Using Optimisation Algorithms", *Applied Energy*, vol. 305, pp.117894, 2022.

  This paper is associated with Chapter 2.

- X. Yu, Y. Wang, L. Zhu, D. Filev and X. Yao, "Engine Calibration with Surrogate-Assisted Bilevel Evolutionary Algorithm", *IEEE Transactions on Cybernetics*, early access, May 1, 2023. DOI: 10.1109/TCYB.2023.3267454.

  This paper is associated with Chapter 4.

- X. Yu, Y. Wang, L. Zhu, D. Filev and X. Yao, "Experience-Based Evolutionary Algorithms for Very Expensive Optimisation", under review.

  This paper is associated with Chapter 5.

Referred Conference Paper

- X. Yu, X. Yao, Y. Wang, L. Zhu, and D. Filev, "Domination-Based Ordinal Regression for Expensive Multi-Objective Optimisation," in Proceedings of the *2019 IEEE*

Symposium Series on Computational Intelligence (SSCI'19), 2019, pp. 2058-2065.

This paper is associated with Chapter 3.

- X. Yu and X. Yao, "Ordinal Regression Evolutionary Algorithm with Adaptive Sampling Strategy for Expensive Multi-Objective Optimisation", under review.

  This paper is associated with Chapter 3.

Chapter 2

Background and Literature Review

This chapter introduces the background knowledge of SAEAs and reviews research studies related to this thesis. Section 2.1 provides some basic definitions and formulations about the optimisation problems to be solved in this thesis. In Section 2.2, an introduction to SAEAs is given, including the workflow of SAEAs, and the surrogates and management strategies used in SAEAs. Section 2.3 reviews the existing SAEAs designed for solving different expensive optimisation problems, which are related to the work in this thesis. Section 2.4 discusses the details of surrogate modelling methodologies used in the thesis. A chapter summary is presented in Section 2.5.

2.1 Formulation of Computational Expensive Optimisation Problems

Computationally expensive optimisation problems are often encountered in numerous real-world applications [116]. Many of these problems belong to black-box optimisation as they are so sophisticated such that their closed-form expressions are not available. Thus, traditional mathematical optimisation methods cannot be used to solve these problems. To solve many expensive optimisation problems in real-world applications, potential solutions

are evaluated on some industrial facilities. However, the cost of running these industrial facilities is not trivial, so the number of allowed evaluations is limited. This section formulates the expensive optimisation problems we considered in the thesis.

The cost of performance evaluations is the utmost distinction between expensive optimisation problems and non-expensive optimisation problems. Evaluating the performance of an arbitrary solution on an expensive optimisation problem can be time-consuming and/or costly to finance. However, such a distinction cannot be presented in the formulation of expensive optimisation problems. Therefore, the formulation of expensive optimisation problems is equivalent to the formulation of non-expensive optimisation problems. Considering the diversity of expensive optimisation problems, it is unrealistic to formulate all possible expensive optimisation problems in this thesis. As a result, this section only formulates two kinds of expensive optimisation problems mentioned in Section 1.2:

1. Expensive multi-objective optimisation problems (EMOPs).

2. The expensive constrained optimisation problems (ECOPs) with one objective.

Without loss of generalisation, this thesis assumes that all optimisation problems discussed in the thesis aim to minimise the objective(s). Then the EMOP can be formulated as follows:

Problem 2.1.1 ***Expensive Multi-Objective Optimisation Problem****:*
For a given evaluation budget FE_{max}, obtain the Pareto set for the following multi-objective optimisation problem:

$$\underset{\boldsymbol{x}\in X}{minimise} \quad f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$$

where f_i denotes the i^{th} objective function and m denotes the number of objectives, X is the decision space of the problem. In other words, X is the domain of optimisation problem f.

Pareto set and Pareto front are defined as follows:

Definition 2.1.1 ***Pareto dominance****:*

A solution $\boldsymbol{x}^1$ *is said to dominate another solution* $\boldsymbol{x}^2$ *(denoted by* $\boldsymbol{x}^1 \prec \boldsymbol{x}^2$*) if and only if:*

$$\forall k \in \{1, 2, \ldots, m\} : f_k(\boldsymbol{x}^1) \leq f_k(\boldsymbol{x}^2) \wedge$$

$$\exists k \in \{1, 2, \ldots, m\} : f_k(\boldsymbol{x}^1) < f_k(\boldsymbol{x}^2)$$

Definition 2.1.2 ***Non-dominated solution****:*

A non-dominated solution $\boldsymbol{x}^\star$ *in the decision space* X *is a solution that cannot be dominated by any other solutions in* X*:*

$$\nexists \boldsymbol{x} \in X : \boldsymbol{x} \prec \boldsymbol{x}^\star$$

Definition 2.1.3 ***Pareto set****:*

Pareto set S_{ps} *is the set of all non-dominated solutions in the decision space* X*:*

$$S_{ps} = \{\boldsymbol{x}^\star \in X | \nexists \boldsymbol{x} \in X : \boldsymbol{x} \prec \boldsymbol{x}^\star\}$$

Definition 2.1.4 ***Pareto front****:*

Pareto front S_{pf} *is the corresponding unique set of the Pareto set in the objective space:*

$$S_{pf} = \{f(\boldsymbol{x}) | \boldsymbol{x} \in S_{ps}\}$$

The second problem, ECOP, is formulated as:

Problem 2.1.2 ***Expensive Constrained Single-Objective Optimisation Problem****:*

For a given evaluation budget FE_{max}*, obtain the minimum for the following constrained optimisation problem:*

$$\begin{aligned} \underset{\boldsymbol{x} \in X}{\textit{minimise}} \quad & f(\boldsymbol{x}) \\ \textit{subject to} \quad & g_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, p \\ & k_i(\boldsymbol{x}) = 0,\ i = 1, \ldots, q. \end{aligned}$$

where f, g_i, k_j denotes the objective, inequality constraints, and equality constraints, respectively. p and q are the number of inequality and equality constraints, respectively. X is the decision space of the problem.

2.2 Surrogate-Assisted Evolutionary Algorithms

SAEAs have been widely used to solve expensive optimisation problems in the literature. To understand the working mechanism of SAEAs and how can SAEAs solve expensive optimisation problems, this section introduces the background knowledge about SAEAs.

2.2.1 General Workflow of SAEAs

A general workflow of SAEAs is illustrated in Fig. 2.1. Initially, a set of solutions are sampled

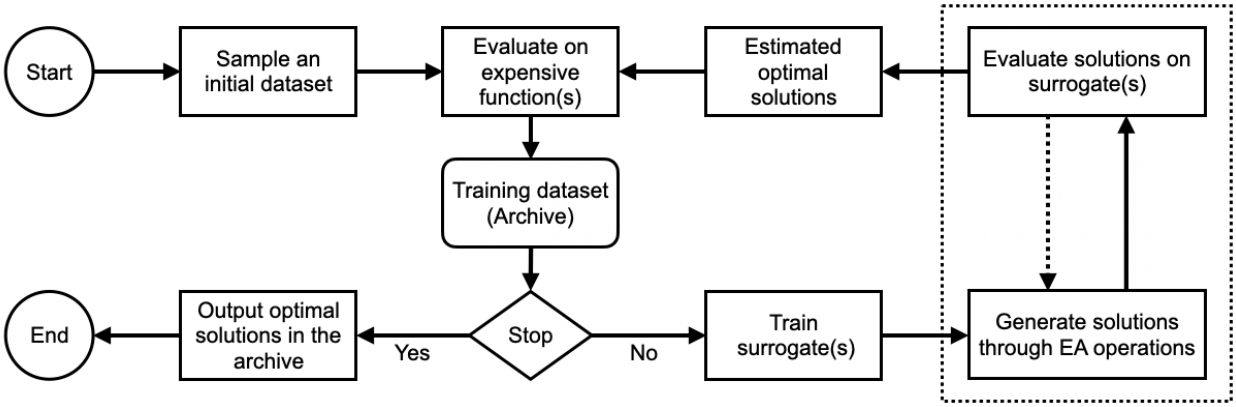


Figure 2.1: A general workflow of surrogate-assisted evolutionary algorithms.

from the decision space through some design of experiment (DoE) methods. The sampled dataset is evaluated on the expensive function and then saved as an archive. The archive serves as a training dataset and is used to train computational efficient surrogates. Then, new offspring solutions are generated through running EAs or conducting some reproductive operations such as crossover and mutation. The generated solutions are evaluated on surrogates and some estimated optimal solutions are selected based on some infill criteria. These

selected solutions are evaluated on expensive functions to obtain their real performance, the evaluated solutions are added to the archive for further surrogate training. If the budget of expensive evaluations has run out, then the surrogate-assisted evolutionary optimisation will stop and the optimal solutions in the archive will be outputted as optimisation results. Otherwise, the evolutionary optimisation will continue and new offspring solutions will be generated.

2.2.2 Issues in SAEAs

This subsection discusses some important issues existing in SAEAs separately.

Initial dataset

The initial dataset is used to initialise surrogates. The dataset should be well-distributed in the decision space to ensure a global exploration of the whole decision space. Many DoE methods have been widely used to sample the initial dataset, such as Latin hypercube sampling [60], uniformly sampling, and randomly sampling. There has no guideline for selecting DoE methods. Many studies in the literature use Latin hypercube sampling due to its popularity [68, 115, 90]. However, one should make sure that the selected DoE method is applicable for the problem to be solved. For example, Latin hypercube sampling is not applicable for expensive combinatorial optimisation problems, thus random sampling is used in [104]. Note that there is not a clear requirement for the size of the initial dataset. Conventionally, 10 d or 11d-1 solutions are sampled as the initial dataset [46, 47, 120, 68, 115], where d is the dimension of the decision space. Recently, an initial dataset of size $3d$ or $4d$ has been demonstrated to be efficient for some single-objective optimisation problems [100]. Besides, the evaluation budget should be considered as the size of the initial dataset must be fewer than the number of allowed evaluations.

Surrogates

For the sake of saving expensive function evaluations, the surrogates used in SAEAs are designed to mimic the behaviour of expensive functions, then a majority of expensive function evaluations can be replaced with surrogate evaluations.

Surrogate categories: Existing surrogates can be mainly divided into two categories based on the purpose of the surrogate modelling. In the first category, surrogates aim to provide accurate approximations of expensive functions. For a given input, it is desirable for surrogates to output the exact same results as querying the expensive functions. Typically, such surrogates are implemented using regression-based models. Instead of approximating exact function values, the surrogates in the second category are trained from the perspective of assisting evolutionary optimisation directly. These surrogates are different from the expensive functions, but they have the same preference for solutions as expensive functions. Usually, they are trained to predict which solutions are preferable to other solutions [31], or are trained to predict if the given solutions are feasible [82, 30]. In the literature, both regression-based models and classification-based models have been employed to implement this kind of surrogate.

Surrogate modelling techniques: The technique of surrogate modelling is also an issue. Plenty of models have been employed as surrogates in SAEAs, such as Kriging model (also known as Gaussian processes) [46, 47, 115, 90], neural network [64, 72, 117, 31], polynomials (also known as response surface methodologies) [36, 88], support vector machine (SVM) [1] and some ensemble surrogates [28]. However, there has very little guideline about how to choose the model for building surrogates, and a model is selected due to either its popularity or its usage in the domain of the problems to be solved [9]. For example, some models are selected because they give uncertainty estimations which are critical to some surrogate modelling techniques. In general, the Kriging model is the most popular one among the aforesaid models because it is able to provide uncertainty information in the

prediction. When uncertainty information is available, more flexible surrogate management strategies can be designed to reach a trade-off between exploration and exploitation during the surrogate-assisted optimisation.

Surrogate Management

Surrogate management, also known as evolution control, is the strategy to manage surrogates in SAEAs. A surrogate management strategy covers several issues that should be of concern, such as how to use surrogate evaluations to select solutions for expensive evaluations, and how to update surrogates with evaluated solutions. Specifically, the selection of solutions for expensive evaluations is the most important issue in a surrogate management strategy (see the modules included in the dashed block in Fig. 2.1), it consists of two parts: the framework of selection and the criterion of selection.

Selection frameworks: The selection framework describes how to use surrogate evaluations to select solutions for expensive evaluation. Existing surrogate management strategies can be mainly classified into two categories based on their selection frameworks: generation-based evolution control and individual-based evolution control [45]. In generation-based evolution control, an EA is applied to generate offspring solutions. During the evolutionary optimisation, all offspring solutions in a generation are evaluated using either expensive functions or approximation surrogates, the frequency of using expensive function evaluations can be fixed [21] or adaptive [66]. Note that if one generation of offspring solutions are evaluated on surrogates, then the next generation of offspring solutions will be generated based on the evaluation results on surrogates (see the dashed arrow in Fig. 2.1). Another way to implement generation-based evolution control is to run a complete EA on approximation surrogates. The output of this EA is an optimum on surrogates in terms of estimated objective values or other acquisition functions such as expected improvement (EI) [46]. The optimum is selected as an estimated optimal solution and re-evaluated on expensive functions. For instance, ParEGO [47] employs a steady-state EA to search for the

solution with maximum for re-evaluation. By comparison, individual-based evolution control generates a generation of offspring solutions through crossover and mutation operations. All solutions in the generation are firstly evaluated on surrogates and then some of them are chosen for re-evaluation on expensive functions. Compared with generation-based evolution control, the individual-based one is more flexible when selecting solutions for expensive function evaluations. In contrast, generation-based evolution control tends to be preferable to the individual-based one if the EA is implemented in parallel [43].

Infill criteria: The infill criterion is an environmental selection criterion used to select optimal solutions. Given a generation of offspring solutions and their evaluation results on surrogates, a SAEA needs an infill criterion to define which solutions are optimal and thus should be selected. Many infill criteria have been developed in the literature. SAEAs often use surrogate prediction as their infill criterion if their surrogates provide only one prediction. For instance, CSEA [68] employs a neural network as its surrogate, such a surrogate predicts the probability that a given solution is non-dominated by reference points. In the process of generation-based evolution control, the solutions with high probabilities to dominate reference points are selected for the reproduction of the next generation. And in the last generation, the offspring solutions whose probabilities are higher than 0.9 are selected for re-evaluation on expensive functions. Some infill criteria also take uncertainty into account. The uncertainty can be measured as the distance between the solution to be evaluated and the evaluated solution in the archive [4]. A more straight forward way to obtain uncertainty is to use the surrogates which are able to provide both prediction and uncertainty, such as Kriging model [109]. Infill criteria such as EI [46, 120], lower confidence bound [71, 56], and probability of improvement (PoI) [122, 11] have been widely used in SAEAs when surrogates can provide uncertainty. Additionally, to explore decision space and maintain the diversity of the selected solutions, some SAEAs attempt to select representative solutions through using clustering-based infill criteria. For example, in MOEA/D-EGO [120], offspring solutions that have been evaluated on surrogates are divided into several clusters firstly, then the solutions

with maximum EI in each cluster are selected as optimal ones for expensive evaluations.

Surrogate update: Surrogates should be updated to maintain their quality during evolutionary optimisation. Existing SAEAs usually update their surrogates when new evaluated solutions are added to the archive, which is beneficial to the accuracy of surrogate approximations. However, as the number of evaluated solutions increases, the size of the archive will be larger than before, which results in an increase in the training time of surrogates. To reduce the time cost of surrogate training without degrading the frequency of surrogate updates, some SAEAs use a small dataset to train their surrogates [47, 120]. Hence, how to select a subset of the archive as the training dataset is also an issue in surrogate management. The solutions in the training dataset can be chosen based on randomness, fitness, or the distribution in the decision space. For instance, in [47], half solutions in the training dataset are selected from the archive on the basis of their aggregated fitness, while the remaining solutions are selected from the archive randomly. In [120], training datasets are generated through grouping the archive into several clusters. Apart from the consideration for the training time, the demand for local approximations is another reason to train surrogates with a subset of the archive [106]. In this situation, the solutions that are close to the areas of interest will be added to the training dataset.

Evolutionary algorithms

A variety of EAs and reproductive operators have been used in SAEAs to generate candidate solutions, including conventional EAs such as genetic algorithm (GA) [33, 34], evolutionary programming (EP) [24], evolutionary strategy (ES) [84, 3], genetic programming (GP) [49], differential evolution (DE) [92] and swarm intelligence algorithms such as ant colony optimisation (ACO) [10], and particle swarm optimisation (PSO) [20]. There is little guideline about how to choose EAs for SAEAs since no EA outperforms other EAs significantly. The selection of EAs tends to be arbitrary in many SAEA studies. However, the performance of SAEAs will be affected by the parameter setups of the EAs used in SAEAs, thus proper EA

parameter setups should be considered.

2.3 Literature Review on SAEAs

Based on the background knowledge of SAEAs discussed in the last section, this section further reviews the SAEA studies that are related to the expensive optimisation problems to be solved in this thesis.

2.3.1 SAEAs designed for Expensive Multi-Objective Optimisation

Various multi-objective SAEAs have been proposed to solve EMOPs in past decades. Based on the strategy to handle multiple objectives, these SAEAs can be roughly classified into two categories: conversion-based SAEAs and dominance-based SAEAs. The former uses different conversion techniques to solve EMOPs with some single-objective optimisation methods, while the latter solves EMOPs through dominance relation.

Conversion-based SAEAs extend the idea of single-objective optimisation to the domain of multi-objective optimisation. There are many ways to implement the conversion between single-objective optimisation and multi-objective optimisation. Using a dynamic scalarising weight vector is the simplest way to convert multiple objectives into an aggregate objective [47]. The dynamic scalarising weight vector is changing at each iteration, thus an approximation of Pareto front can be obtained gradually through employing an EA to optimise the aggregate objective function. Decomposition is another popular way to implement the conversion. Decomposition-based methods use diverse static weight vectors to decompose a multi-objective optimisation problem, which results in series of single-objective optimisation subproblems [120]. Different from the use of a dynamic weight vector where only the aggregate objective needs to be optimised, in decomposition-based methods, EAs are employed to optimise all these subproblems in every iteration. For the two kinds of conversion-based SAEAs mentioned above, their solution diversity is dependent on the dis-

tribution of weight vectors. Additionally, performance indicators that are designed for multi-objective optimisation, such as hypervolume (HV) [125], are also used to convert multiple objectives into a single objective [71, 70]. One shortcoming of such indicator-based SAEAs is that the computation of HV or other performance indicators is time-consuming, which limits the application of this kind of SAEAs.

Compared with conversion-based SAEAs, dominance-based SAEAs are predominant in the literature [9] and they do not need conversions. Typically, these SAEAs approximate each objective with one surrogate separately, which makes it convenient for these SAEAs to combine existing MOEAs with the surrogates discussed in Section 2.2.2. For example, NSGA-II [16], a classic dominance-based MOEA, cooperates with a neural network surrogate in NSGA-II-ANN [65]. A reference-guided MOEA, namely RVEA [7], is combined with Kriging surrogates in K-RVEA [8]. And a two-archive assisted MOEA, TA2 [103], is combined with Kriging surrogates in KTA2 [90], where one of two archives is designed for maintaining the diversity of non-dominated solutions in the objective space. However, three drawbacks of approximating each objective with one surrogate should be noted. Firstly, the time complexity of maintaining surrogates grows exponentially with the number of objectives. Second, the cumulative approximation errors from all surrogates will adversely affect the overall approximation accuracy. Thirdly, correlations between objectives are lost.

Recently, a new class of dominance-based SAEAs is proposed to use a single classification-based surrogate to directly learn the dominance relation for solutions. These classification-based surrogates will not suffer from the cumulative errors encountered by multiple regression-based surrogates. For instance, CSEA [68] trains a neural network to predict whether candidate solutions can be dominated by given reference points or not. Such a dominance-based SAEA is efficient since the algorithm needs to maintain only one classification-based surrogate. Apart from learning relations between solutions and reference points, some dominance-based SAEAs also use classification-based surrogates to learn the dominance relations for pairwise solutions. θ-DEA-DP [117] uses two neural networks to predict the Pareto dom-

inance relation and the θ-dominance relation between two solutions, respectively. REMO [31] employs a neural network to fit a ternary classifier, which is able to learn the dominance relation between pairs of solutions. These SAEAs take advantage of learning dominance relations and have found their way into solving EMOPs. But it should be pointed out that the learning of pairwise dominance relations [117, 31] indicates an exponential increase in the complexity of surrogate training and prediction. For relations between reference points and solutions [68], the solutions classified into the same category are not comparable.

2.3.2 SAEAs Designed for Expensive Constrained Optimisation

Many studies have been conducted to solve ECOPs with SAEAs. In [75, 100], radial basis function (RBF) surrogates were used to approximate each constraint separately, which assists the proposed evolutionary programming (EP) algorithm to solve high-dimensional ECOPs. In [44], each nonlinear constraint was approximated by an artificial neural network (ANN) surrogate, these surrogates were assembled into a stochastic ranking evolution strategy (SRES) [81]. Surrogates such as k-nearest neighbour (KNN) regression surrogate [82] and support vector machine (SVM) classification surrogate [30] have been employed to define solution feasibility directly. SA-DECV [62] uses one unique surrogate to approximate both the objective value and the feasibility of solutions. Moreover, adaptive surrogates [87] and the combination of global and local surrogates [106] have been developed to solve ECOPs. Besides, SAEAs have been used to solve diverse ECOPs including multi-objective problems [76], [12], combinatorial problems [104], high-dimensional problems [22], and the ECOPs with inequality constraints [106, 50], equality constraints [112], and nonlinear constraints [44], [121]. In [107], distributed ECOPs were studied where the evaluation of constraints is asynchronous. In [58], a SAEA was designed to solve ECOPs involving mixed-integer variables. From the view of constraint handling techniques (CHTs), the influence of four CHTs on a SAEA was investigated in [63]. A novel CHT was developed in [93] to map the

feasible region into the origin of the Euclidean subspace for ECOPs. A parallel constrained lower confidence bounding approach was proposed to solve ECOPs [6]. In [114], a specific mutation operation was developed to achieve an efficient classification-collaboration between feasible and infeasible solutions.

The aforesaid studies have made achievements of solving ECOPs from different aspects, such as the use of constraint surrogates [77], the category of ECOPs, and the development of CHTs. However, a shortcoming is that the sensitivity of solution feasibility to decision variables has not been investigated in existing studies. Although CHTs such as feasibility rule [13], ϵ-constrained method [94], stochastic ranking [81], and diversity maintenance [61] have been widely used to solve ECOPs, these CHTs tend to pay their attention to every decision variable evenly. In [29], constrained optimisation problems are solved via a bilevel architecture, but the problems are not expensive and every decision variable is optimised in both lower and upper levels. In real-world applications, decision variables may have different impacts on the feasibility of solutions. Hence, it is desirable to allocate optimisation resources on the basis of the impact of variables on solution feasibility. To fill this gap, this thesis analyzes the sensitivity of solution feasibility to decision variables, the decision variables that have a significant impact on solution feasibility will be divided into lower-level variables. A bilevel architecture is employed to handle constraints where more effort is made on lower-level variables to explore the feasible region.

2.3.3 SAEAs Based on Experience Learning and Transformation

Many SAEAs have been developed to solve expensive optimisation problems. These SAEAs are designed to handle different optimisation scenarios such as single-objective optimisation [100], multi-objective optimisation [74], constrained optimisation [114], large-scale optimisation [75], and combinatorial optimisation [104]. Although these SAEAs have reached some achievements in diverse optimisation scenarios, none of them gains experience from past

optimisation problems, making their optimisation start from a zero prior knowledge state. Hence, these SAEAs still need tens to hundreds of evaluated samples to initialise their surrogates due to the demand for reliable surrogates.

In the past decade, experience-based evolutionary optimisation has attracted much attention as it uses the experience obtained from other optimisation problems to improve the optimisation efficiency of target problems, which mimics human capabilities of cognitive and knowledge generalisation [27]. The optimisation problems that provide experience or knowledge are regarded as source tasks, while the target optimisation problems are regarded as target tasks. To obtain useful experience, the tasks that are related to target tasks are chosen as source tasks since they usually share domain-specific features with target tasks. Diverse experience-based evolutionary optimisation methods have been proposed to use the experience gained from related tasks to tackle target tasks. They can be divided into two categories based on the direction of experience transformation.

In the first category, experience is transformed mutually. Every considered optimisation problem is a target task and is also one of the source tasks of other optimisation problems. In other words, the roles of source task and target task are compatible. One representative tributary is EMTO that aims to solve multiple optimisation tasks concurrently [19, 108, 53, 2, 113]. In EMTO, experience is learned, updated, and spontaneously shared among target tasks through multi-task learning techniques. A variant of EMTO is multiforms optimisation [27, 119, 26]. In multiforms optimisation, multi-task learning methods are employed to learn experience from distinct formulations of a single target task.

In the second category, experience is transformed unidirectionally. The roles of source task and target task are not compatible, an optimisation problem cannot be a source task and a target task simultaneously. One popular tributary is transfer optimisation which employs transfer learning techniques to transform experience from source tasks to target tasks [95, 41, 40, 102]. In transfer learning, experience can be transformed from a single source task, multiple source tasks, or even source tasks from a different domain [124]. However,

these transfer learning techniques pay more attention to experience transformation instead of experience learning. Although diverse and complex situations of experience transformation have been studied [78, 79], the difficult of learning experience from small (expensive) source tasks has not been well studied. Actually, a common scenario in transfer learning is that the source task(s) is/are large enough such that useful experience can be obtained easily through solving source task(s) [124]. In contrast to transfer optimisation, recently, some experience-based optimisation algorithms attempted to use meta-learning methods to learn experience from small source tasks, which is known as few-shot optimisation [111]. Since meta-learning only works for related tasks in the same domain, the situations of experience transformation are less complex than that of transfer learning, thus meta-learning pays more attention to experience learning instead of experience transformation. Domain-specific features are extracted as experience and no related task needs to be solved.

In this thesis, we consider experience-based optimisation in the context of few-shot problems [96, 111]. That is, we have many related expensive tasks which serve as source tasks and one new expensive task which is our target task, each related task provides a few samples and the target task allows a few fitness evaluations. Our work belongs to the few-shot optimisation in the second category above since our experience is transformed unidirectionally. More importantly, our experience is learned across many related expensive tasks, rather than gained through solving more or fewer source tasks.

Existing studies in few-shot optimisation only work for global optimisation [111], leaving other optimisation scenarios such as multi-objective optimisation and constrained optimisation still awaiting investigation. In addition, in-depth ablation studies are lacking in the literature, it is unclear which factors will affect the performance of few-shot optimisation. Moreover, some studies used existing meta-learning models [69] as their surrogates. No further adaptations are made to these surrogates during the optimisation since they are not designed for optimisation originally. Therefore, a novel meta-learning model for few-shot optimisation is desirable. Our work fills the aforesaid gaps by proposing a novel meta-

learning modelling method and a general experience-based SAEA framework. As a result, accurate surrogates and competitive optimisation results can be achieved while the cost of surrogate initialisation is only $1d$ evaluations for a d-dimensional expensive target problem.

2.4 Surrogate Modelling

This section introduces the surrogate modelling methodologies used in the thesis, including the Kriging model and the feedforward neural network.

2.4.1 Kriging Model

Kriging model, also known as Gaussian process model [46] or design and analysis of computer experiments (DACE) model [83], is a stochastic process model used to approximate an unknown objective function. It plays an important role in this thesis since it is employed to build surrogates in Chapters 3, 4, and 5. To avoid potential confusion and help the understanding of the next three sections, the working mechanism of the Kriging model is described below.

A common way to approximate an unknown objective function with n observations is linear regression:

$$y(\mathbf{x}^i) = \sum_{k=1}^{N} \beta_k f_k(\mathbf{x}^i) + \epsilon^i, \tag{2.1}$$

where $\mathbf{x}^i$ is the i^{th} sample point observed from the objective function. $f_k(\mathbf{x}^i)$, β_k are a linear or nonlinear function of $\mathbf{x}^i$ and its coefficient, respectively. N is the number of functions $f(\mathbf{x})$. ϵ^i is an independent error term, which is normally distributed with mean zero and variance σ^2.

However, a stochastic process model such as Kriging does not assume that the error terms ϵ are independent. Hence, an error term ϵ^i is rewritten as $\epsilon(\mathbf{x}^i)$. Moreover, these error terms are assumed to be related or correlated to each other. The correlation between two er-

ror terms $\epsilon(\mathbf{x}^i)$ and $\epsilon(\mathbf{x}^j)$ is inversely proportional to the distance between the corresponding points [46]. The correlation function in the Kriging model is defined as:

$$Corr(\epsilon(\mathbf{x}^i), \epsilon(\mathbf{x}^j)) = exp[-dis(\mathbf{x}^i, \mathbf{x}^j)], \tag{2.2}$$

where the distance between two points $\mathbf{x}^i$ and $\mathbf{x}^j$ are measured using the special weighted distance formula shown below:

$$dis(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=1}^{d} \theta_i |x_k^i - x_k^j|^{p_k}, \tag{2.3}$$

where d is the number of decision variables, $\boldsymbol{\theta} \in \mathbb{R}^d_{\geq 0}$ and $\mathbf{p} \in [1, 2]^d$ are parameters of the Kriging model. It can be seen from Eq.(2.2) that the correlation is ranged within $(0, 1]$ and is increasing as the distance between two points decreases. Particularly, in Eq.(2.3), the parameter θ_k can be explained as the importance of the decision variable x_k, and the parameter p_k can be interpreted as the smoothness of the correlation function in the k^{th} coordinate direction.

Due to the effectiveness of correlation modelling, the regression model in Eq.(2.1) can be simplified without degrading modelling performance [46]. Clearly, all regression terms are replaced with a constant term, thus the Kriging regression model can be rewritten as follows:

$$y(\mathbf{x}^i) = \mu + \epsilon(\mathbf{x}^i), \tag{2.4}$$

where μ is the mean of this stochastic process, $\epsilon(\mathbf{x}^i) \sim \mathcal{N}(0, \sigma^2)$.

To train the Kriging model and estimate the parameters $\boldsymbol{\theta}, \mathbf{p}$ in Eq.(2.3), the following likelihood function is maximised:

$$\frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|\mathbf{R}|^{1/2}} exp[-\frac{(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}], \tag{2.5}$$

where $|\mathbf{R}|$ is the determinant of the correlation matrix, each element in the matrix is obtained using Eq.(2.2). $\mathbf{y}$ is the n-dimensional vector of dependent variables that observed from the objective function. The mean value μ and variance σ^2 in Eq.(2.4) and Eq.(2.5) can be estimated by:

$$\hat{\mu} = \frac{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}, \tag{2.6}$$

$$\hat{\sigma} = \frac{1}{n}(\mathbf{y} - \mathbf{1}\hat{\mu})^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}). \tag{2.7}$$

For a new solution $\mathbf{x}^*$, the Kriging model predicts the approximation of $\hat{y}(\mathbf{x}^*)$ and the uncertainty $\hat{s}^2(\mathbf{x}^*)$ as follows:

$$\hat{y}(\mathbf{x}^*) = \hat{\mu} + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}), \tag{2.8}$$

$$\hat{s}^2(\mathbf{x}^*) = \hat{\sigma}^2(1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r}), \tag{2.9}$$

where $\mathbf{r}$ is a n-dimensional vector of correlations between $\epsilon(\mathbf{x}^*)$ and the error terms at the training data, which can be calculated via Eq.(2.2). More details about the Kriging model and Gaussian Process can be found in [46, 109].

2.4.2 Feedforward Neural Network

Artificial neural networks are models designed to interact with the real-world objectives in a way inspired by biological nervous systems [48]. They are parallel interconnected networks of transfer functions (neurons) and have hierarchical organisations. Feedforward neural networks (FNNs) are an important tributary of artificial neural networks where connections between the neurons do not form a cycle. A FNN is combined with the Kriging model to build surrogates in Chapter 5, thus this subsection presents the background of FNNs as a preliminary.

The architecture of FNNs is illustrated in Fig. 2.2 through an example of a three-layer

FNN. It can be seen that a k-layer FNN model consists of an input layer, k-1 hidden layers,

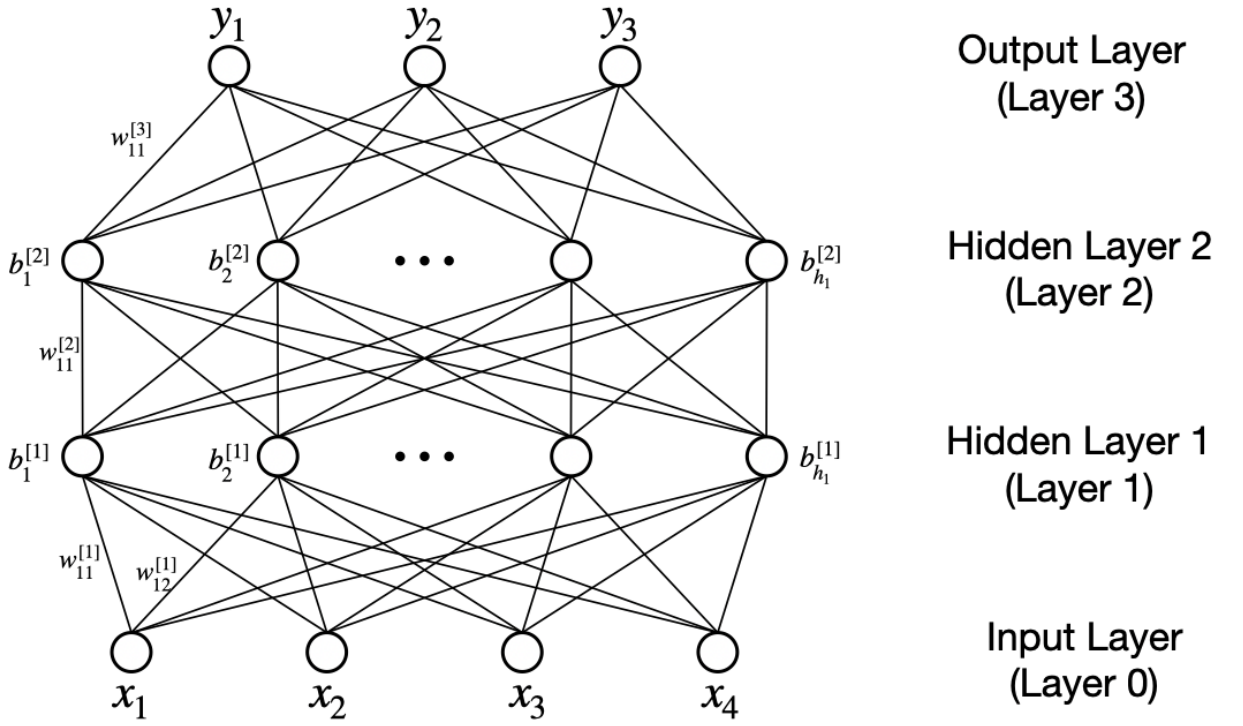


Figure 2.2: Architecture of a three-layer FNN. The input layer does not conduct any computation operations, thus the number of layers k counts only hidden layers and the output layer. In this example, the number of decision variables $d = 4$ and the number of objectives $m = 3$. Therefore, the input layer has 4 neurons and the output layer has 3 neurons. Two hidden layers have h_1 and h_2 neurons, respectively.

and an output layer. The neuron, also known as the unit, is the basic component in FNNs. Each layer of a FNN is made up by several neurons. Specially, the input and output layers contain d, m neurons, respectively, where d is the number of decision variables and m is the number of objectives to approximate. In comparison, the number of neurons in a hidden layer is flexible and always determined based on the demand of modelling complexity. The FNNs with $k > 2$ also known as deep FNNs.

The workflow of a FNN starts from the input layer which receives input data $\mathbf{x}$. For the l^{th} layer ($1 \leq l \leq k$), the outputs of its neurons $\mathbf{a}^{[l]}$ are computed by

$$\mathbf{a}^{[l]} = h(\mathbf{w}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}), \tag{2.10}$$

where $\mathbf{w}^{[l]}$ are the weights of connections between the l^{th} layer and the $(l-1)^{th}$ layer, $\mathbf{b}^{[l]}$ are the biases of neurons in the l^{th} layer. Both $\mathbf{w}^{[l]}$ and $\mathbf{b}^{[l]}$ are the parameters need to be tuned in the l^{th} layer. $\mathbf{a}^{[l-1]}$ are the outputs of the $(l-1)^{th}$ layer and $\mathbf{a}^{[0]} = \mathbf{x}$. h is a nonlinear activation function, e.g. sigmoid, tanh, ReLU [67]. The output layer (the k^{th} layer) gives the output of this FNN: $\mathbf{y} = \mathbf{a}^{[k]}$. The parameters $\mathbf{w} = (\mathbf{w}^{[1]}, \mathbf{w}^{[2]}, \ldots, \mathbf{w}^{[k]})$ and $\mathbf{b} = (\mathbf{b}^{[1]}, \mathbf{b}^{[2]}, \ldots, \mathbf{b}^{[k]})$ are often trained through backpropagation [80].

2.5 Chapter Summary

In this chapter, the background knowledge related to this thesis has been given. Firstly, basic definitions and mathematical formulations of expensive optimisation problems are introduced to clarify what optimisation problems we are solving in this thesis. Then, a detailed description of SAEAs is given, including the explanation of SAEA working mechanism and some discussions about the important issues in the initial dataset, surrogates, surrogate management strategies, and EAs. Literature reviews on SAEAs designed for EMOPs and ECOPs as well as experience-based SAEAs are conducted, the gaps in solving expensive optimisation problems with SAEAs are analyzed. Finally, due to the use of the Kriging model in Chapters 3-5, an introduction to the Kriging model is given, which covers the nature of the Kriging model as a stochastic process model, the training methodology of the Kriging model, and the prediction methodology through the Kriging model. A brief introduction to feedforward neural network is also provided since neural networks are related to our work in Chapter 5.

Chapter 3

Ordinal Surrogate Assisted Evolutionary Algorithms for Expensive Multi-Objective Optimisation

This chapter proposes a dominance-based ordinal regression surrogate, in which a Kriging model is employed to learn the dominance-based relation values and to approximate the ordinal landscape of objective functions. Coupling with a hybrid surrogate management strategy, the solutions with higher probabilities to dominate others are selected and evaluated on expensive objective functions. Moreover, an adaptive surrogate management strategy is proposed to further improve the optimisation performance for EMOPs. The adaptive management strategy analyzes the state of optimisation and the distribution of non-dominated solutions. Based on the analysis result, the management strategy will adapt the number of solutions sampled from global search, local search, or diversity maintenance, leading to a dynamic balance between convergence and diversity. Our computational experiments on DTLZ benchmark test problems have shown that our algorithms outperform all compared state-of-the-art SAEAs. The effectiveness of our ordinal surrogate and two surrogate management strategies has also been demonstrated.

3.1 Motivation

Existing SAEAs that are designed for EMOPs can be mainly classified into two categories based on their purposes of using surrogates. In most existing SAEAs, surrogates aim to approximate the fitness of solutions or the objective functions of expensive problems [42]. Therefore, regression-based models are used as surrogates in these SAEAs. For example, ParEGO [47] employs a Kriging model to estimate the fitness of a weighted objective function. MOEA/D-EGO [120] uses multiple Kriging models to approximate all objective functions. In K-RVEA [8] and KTA2 [90], Kriging models are also trained to do fitness regression. However, from the perspective of assisting evolutionary optimisation, what MOEAs need is the relation between solutions rather than accurate fitness values. As a result, some recently proposed SAEAs use classification-based surrogates to learn the relation between solutions directly. CSEA [68] trains a neural network to predict whether candidate solutions can be dominated by given reference points or not. θ-DEA-DP [117] uses two neural networks to predict the Pareto dominance relation and the θ-dominance relation between two solutions, respectively. REMO [31] employs a neural network to fit a ternary classifier, which is able to learn the dominance relation between pairs of solutions. Compared with regression-based SAEAs, although classification-based SAEAs take advantages of learning solution relations directly, their drawbacks are obvious: The learning of pairwise relations [117, 31] indicates an exponential increase in the complexity of surrogate training and prediction. For relations between reference points and solutions [68], the solutions classified into the same category are not comparable.

Considering the shortcomings of regression-based and classification-based surrogates, it is desirable to develop a novel surrogate model. Such a surrogate should be able to learn the ordinal relation between solutions directly with a low training complexity. In addition, most solutions should be comparable based on their surrogate evaluation results. Besides, to improve the optimisation performance on EMOPs, an efficient surrogate management

strategy should be developed. We firstly develop a hybrid surrogate management strategy to balance convergence and diversity. Then, an adaptive surrogate management strategy is proposed to balance convergence and diversity in a more flexible way, which further improves the performance of our SAEA.

3.2 Ordinal Regression Evolutionary Algorithm (OREA)

This chapter firstly propose a novel SAEA, namely OREA, to solve EMOPs with the ordinal-regression-based surrogate.

3.2.1 General Framework

The framework of OREA is depicted in Algorithm 1, it can be divided into the following steps [1]:

1. Initialisation: An initial dataset of size $11d$ - 1 is sampled from the decision space using the Latin hypercube sampling (LHS) [60] (line 1), where d is the dimensionality of decision variables. The sampled solutions are evaluated on expensive functions f and then saved in an archive S_A (line 2). A group of reference vectors is generated using the method in [51] (line 3).

2. Ordinal relation quantification and surrogate modelling: The domination-based ordinal relations are quantified (line 5) and then used to train the ordinal-regression-based surrogate h (line 6).

3. Reproduction: A generation-based evolution control method is used to produce one solution $\mathbf{x}^{1*}$ (line 7), and an individual-based evolution control method is used to produce another solution $\mathbf{x}^{2*}$ (line 8).

[1] Equations in lines 2, 10, and 11 denote assignment operations, left arrows in lines 6, 7, and 8 indicate results of invoking functions.

Algorithm 1 OREA framework

Input:

f: Objective functions of the expensive optimisation problem;
FE_{max}: Maximum number of allowed fitness evaluations;

Procedure:

1: Sample a set of solutions $\{\mathbf{x}^1, \ldots, \mathbf{x}^{11d-1}\}$ and evaluate them on f.
2: Save all evaluated solution $(\mathbf{x}, f(\mathbf{x}))$ in an archive S_A. Set the number of used evaluations $FE = |S_A|$. /*$|S_A|$ is the size of S_A.*/
3: Generate reference vectors RV.
4: **while** $FE < FE_{max}$ **do**
5: /*Ordinal Relation Quantification and Surrogate Modelling*/
Quantified relation values $S_V \leftarrow Quantification$. /*See Algorithm 2.*/
6: Train the ordinal surrogate $h \leftarrow$ Kriging(S_A, S_V).
/*Reproduction with Hybrid Management Strategy*/
7: $\mathbf{x}^{1*} \leftarrow$ Gen_Reproduce(h).
8: $\mathbf{x}^{2*} \leftarrow$ Ind_Reproduce(h, RV, S_A, S_V).
/*Update*/
9: Evaluate new solutions $\mathbf{x}^{1*}$ and $\mathbf{x}^{2*}$ on expensive functions f.
10: Update $S_A = S_A \cup \{(\mathbf{x}^{1*}, f(\mathbf{x}^{1*})), (\mathbf{x}^{2*}, f(\mathbf{x}^{2*}))\}$.
11: $FE = FE + 2$
12: **end while**

Output:

Non-dominated solutions in archive S_A.

4. Update: New solutions are evaluated on the expensive functions (line 9), archive S_A is updated (line 10) and the number of used function evaluations FE is updated (line 11).

The details of these steps are explained in the following subsections.

3.2.2 Ordinal-Regression-Based Surrogate

In the ordinal regression, the domination-based ordinal relations between solutions $\mathbf{x}$ and a set of reference points S_{RP} are quantified and then used to fit a regression-based Kriging model. Considering the Kriging model is a common modelling method that has been widely used in many studies and its details can be found in [109], in this section, we focus on the quantification of domination-based ordinal relations (line 5 in Algorithm 1). The quantifi-

cation consists of two phases: the generation of reference and the computation of relation values. The pseudo code is given in Algorithm 2.

Algorithm 2 Quantification (S_A, n_o, c_{shape})

Input:

$S_A = \{S_{\mathbf{x}}, S_{f(\mathbf{x})}\}$: An archive of evaluated solutions and their fitnesses;
n_o: Minimum of ordinal levels;
c_{shape}: A coefficient for shaping reference points.

Procedure:

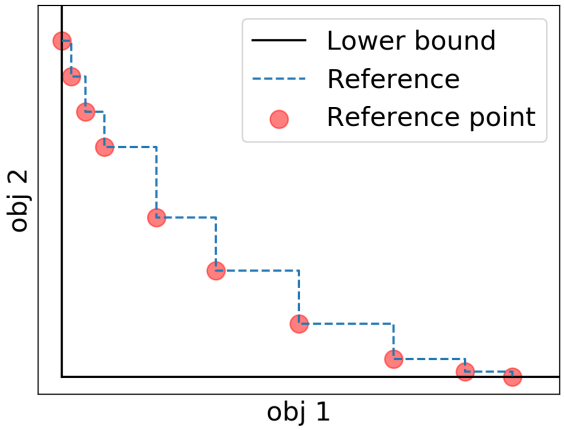
Attainment front $S_{AF} \leftarrow$ Non-dominated solutions in $S_{f(\mathbf{x})}$.
$\mathbf{b}_u, \mathbf{b}_l \leftarrow$ Upper and lower bounds of S_{AF}.
$S_{AF_0} = S_{AF} - \mathbf{b}_l$. /*Initial reference points.*/
/*Generation of Reference*/
Set the size of reference points $|S_{RP}| = 0$.
while $|S_{RP}| < 2$ **do**
  Shape criterion $C_{shape} = (\mathbf{b}_u - \mathbf{b}_l) \times c_{shape}$.
  $S_{RP} \leftarrow$ Exclusion(S_{AF_0}, C_{shape})
  $c_{shape} = c_{shape} - 0.01$
end while
Update S_{RP} to shape the reference, making the reference interact with $\mathbf{b}_l$.
for $f_0(\mathbf{x}^i) \in S_{AF_0}$ but $\notin S_{RP}$ **do**
  $S_{RP} \leftarrow S_{RP} \cup f_0(\mathbf{x}^i)$ if $\nexists \mathbf{x} \in S_{RP} : \mathbf{x} \prec f_0(\mathbf{x}^i)$. /*$f_0(\mathbf{x}) = f(\mathbf{x}) - \mathbf{b}_l$.*/
end for
/*Computation of Relation Values*/
$S_{f_0(\mathbf{x})} = S_{f(\mathbf{x})} - \mathbf{b}_l$
Compute extension coefficient $ec(\mathbf{x})$ for all $f_0(\mathbf{x}) \in S_{f_0(\mathbf{x})}$ but $\notin S_{RP}$, results in a set S_c.
Number of ordinal levels $N_o \leftarrow \max(n_o, \lceil \frac{|S_A|}{|S_{RP}|} \rceil)$.
Solutions in the first ordinal level: $S_1 = S_{RP}$.
$\{S_2, \ldots, S_{N_o}\} \leftarrow$ Divide remaining solutions $\mathbf{x} \in S_A$ into $N_o - 1$ ordinal levels uniformly based on their $ec(\mathbf{x})$.
Quantified relation value for the i^{th} ordinal level $S_V(S_i) = \frac{N_o - i}{N_o - 1}$, $i = 1, \ldots, N_o$.

Output:

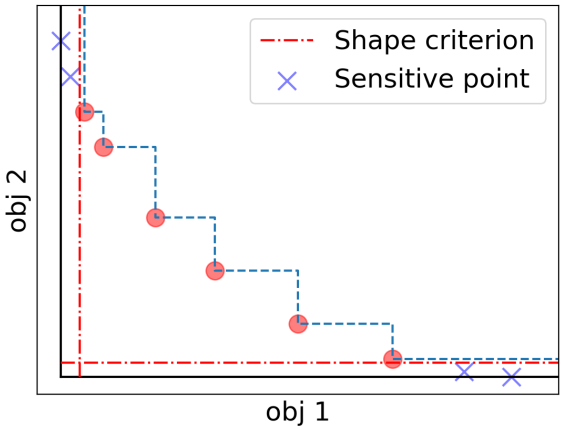
S_V: A set of quantified relation values for solutions in S_A.

Generation of Reference

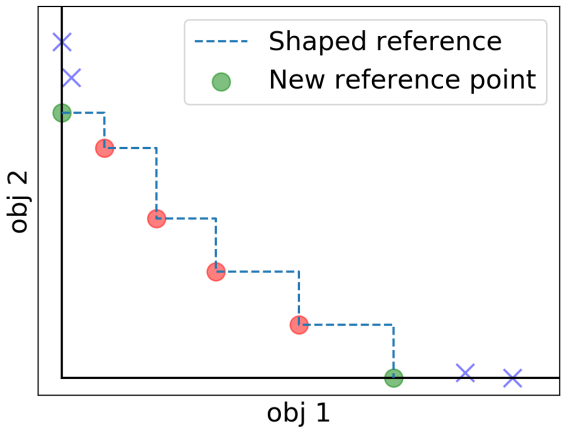
The reference points for quantification and the quantified relation values are updated iteratively during the evolutionary optimisation (Algorithm 1). For an arbitrary iteration of Algorithm 1, the procedure of generating reference is illustrated in both Fig. 3.1 and Algo-



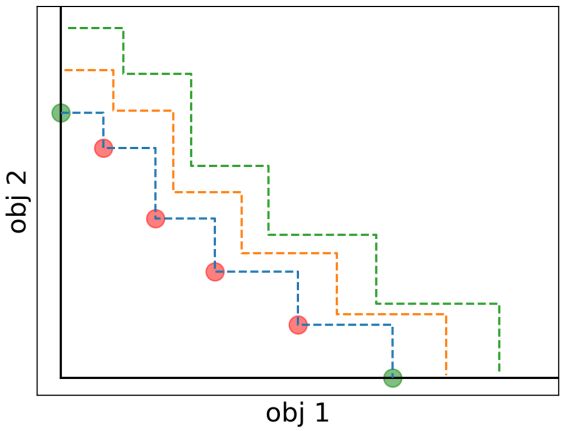
(a) Initial reference



(b) Shape lines and sensitive points



(c) Shaped reference



(d) Extended reference

Figure 3.1: The reference used to quantify the domination-based ordinal relations between solutions. In Fig. (a), the initial reference is the centralised attainment front, a boundary consist of the non-dominated solutions found so far. The initial reference points are these centralised non-dominated solutions. In Fig. (b), sensitive points that are close to the lower bound of the current reference will be identified and excluded. In Fig. (c), a shaped reference is formed. In Fig. (d), the shaped reference is extended to quantify the domination-based ordinal relations between solutions.

rithm 2. First, the reference points S_{RP} are initialised using the non-dominated solutions in archive. Based on the domination-based relation between S_{RP} and other solutions, an attainment front S_{AF} is obtained and then centralised as the initial reference S_{AF_0} (lines 1-3,

see Fig. 3.1a). Second, the sensitive solutions in S_{AF_0} (crosses in Fig. 3.1b) are excluded from S_{RP} (lines 4-9). For these sensitive solutions, at least one of their fitnesses is close to the lower bound of S_{AF}. Compared with other non-sensitive solutions in S_{AF_0}, despite that the fitnesses of sensitive solutions in some objectives can be much greater than the fitnesses of non-sensitive solutions, these sensitive solutions are still non-dominated. It is obvious that these sensitive solutions are not desirable as other non-sensitive solutions in S_{AF_0}. In practice, let $\mathbf{b}_u$ and $\mathbf{b}_l$ denote the upper bound and lower bound of S_{AF}, respectively. Then the shape criterion is defined as the product of the range of objective values and a shape coefficient c_{shape}:

$$C_{shape} = (\mathbf{b}_u - \mathbf{b}_l) \times c_{shape}. \tag{3.1}$$

All solutions $\mathbf{x} \in S_{AF_0}$ that are located between $\mathbf{b}_l$ (black solid lines in Fig. 3.1b) and the shape criterion (red dash lines in Fig. 3.1b) are identified as sensitive points. Third, new reference points that are derived from the upper bound of S_{RP} (green points in Fig. 3.1c) are added to S_{RP}, making the reference interact with the lower bound $\mathbf{b}_l$ (line 10). Additionally, some sensitive points are non-dominated with respect to the updated S_{RP} but were excluded in the second step, these solutions are added back to S_{RP} (lines 11-13). The selected reference points S_{RP} construct a shaped reference in Fig. 3.1c.

Computation of Relation Values

The shaped reference in Fig. 3.1c is extended to quantify ordinal relations for the purpose of ordinal regression (see Fig. 3.1d). As the reference extends, a solution $\mathbf{x} \in S_A$ which was dominated by S_{RP} would be non-dominated with respect to the extended S_{RP}. Hence, we define a domination-based distance metric, namely extension coefficient ec, as follows:

$$ec(\mathbf{x}) = \arg\min_{ec \geq 1} f_0(\mathbf{x}) \prec \{S_{RP} \times ec\}, \tag{3.2}$$

where $f_0(\mathbf{x})$ is the objective values $f(\mathbf{x})$ minus $\mathbf{b}_l$ (line 14). Note that although the extension coefficient $ec(\mathbf{x})$ quantifies the distance between a solution $\mathbf{x}$ and reference S_{RP}, it has not been used to train the ordinal-regression-based surrogate directly. Despite that the lower bound of extension coefficients is fixed to 1.0, their upper bound is dependent on the problem to be optimised and the state of evolutionary optimisation. If these extension coefficients are used to fit the ordinal-regression-based surrogate directly, large fluctuations in extension coefficients will occur between two consecutive iterations. To generate a stable ordinal-regression-based surrogate, relation values are used to fit the surrogate, which are numerical values ranged from 1.0 to 0.0. In detail, solutions in S_A are divided into N_o ordinal levels:

$$N_o = max(n_o, \frac{|S_A|}{|S_{RP}|}), \tag{3.3}$$

where n_o is the minimum of ordinal levels, a hyper-parameter defined in Algorithm 2. The solutions in S_{RP} belong to the first ordinal level, thus a relation value $v_1 = 1.0$ is assigned to them. Remaining solutions in S_A are sorted by their extension coefficients $ec(\mathbf{x})$ and then divided into N_o - 1 ordinal levels uniformly. A smaller $ec(\mathbf{x})$ indicates a smaller ordinal level. The relation value $v_i = \frac{N_o - i}{N_o - 1}$ will be assigned to the solutions $\mathbf{x}$ in the i^{th} ordinal level.

It is noticeable that $\mathbf{b}_u$ and $\mathbf{b}_l$ can be equal in one or more objectives. In such case, these objectives are treated as invalid objectives, they will be ignored during the quantification process. Another extreme condition is that S_{AF} contains only one solution, which means the optimum $f(\mathbf{x}^*)$=$\mathbf{b}_l$ and S_{RP}=$f_0(\mathbf{x}^*)$=$\{(0, 0, \ldots, 0)\}$. In this case, the objective space will be divided evenly to form the ordinal levels.

Comparison between OREA and PRL

It is noticeable that PRL [85] employed a Pareto rank based surrogate, which is similar to the proposed domination-based ordinal-regression-based surrogate at first glance. However, the differences can be summarised as follows:

1. The number of ranks/levels: In PRL surrogate, the number of Pareto ranks, K, depends on how many non-dominated solution groups exist in the archive, while the number of ordinal levels n_o in the ordinal-regression-based surrogate can be pre-defined or self-adaptive.

2. The number of solutions in each rank/level: The number of solutions in each Pareto rank group is beyond control, but in the ordinal-regression-based surrogate, every ordinal level (except the first level) shares the same number of solutions.

3. The first rank/level: The first rank group in PRL surrogate is the original archived PF, while the first ordinal level in the ordinal-regression-based surrogate is a shaped one.

4. The output of the surrogates: PRL surrogate predicts the Pareto rank r for given solutions, where $r \in \mathbf{Z}$ and $r < K$. In contrast, the ordinal-regression-based surrogate predicts the mean and variance of domination-based ordinal relation values.

3.2.3 Hybrid Surrogate Management Strategy

In OREA, a hybrid surrogate management strategy is developed to select candidate solutions for expensive evaluation. Such a management strategy is derived from generation-based evolution control and individual-based evolution control. The former is responsible for searching global optima, while the latter uses a local search to maintain the diversity of non-dominated solutions in S_A. For both evolution control methods, their infill sampling criterion is the expected improvement [46], which can be explained as the expectation of locating at the first ordinal level in the ordinal regression:

$$E[S_V(\mathbf{x}^*) \geq 1.0] = (\hat{\mu} - 1.0)\Phi(\frac{\hat{\mu} - 1.0}{\hat{\sigma}}) + \hat{\sigma}\phi(\frac{\hat{\mu} - 1.0}{\hat{\sigma}}), \tag{3.4}$$

where $\hat{\mu}, \hat{\sigma}$ are the surrogate predictions from $h(\mathbf{x}^*)$. Φ and ϕ are cumulative distribution function and probability density function, respectively. Therefore, OREA is also a Bayesian optimisation method.

Generation-Based Evolution Control

The generation-based evolution control employs a PSO to search the candidate solutions on the approximation surrogate h. Clearly, an initial population is sampled from the decision space uniformly, then all the solutions are evaluated on the surrogate until the budget of surrogate evaluations has run out. The result of the PSO run is the final population of offspring solutions, and the best one of them is picked (using the criterion in Eq.(3.4)) as a candidate solution for expensive evaluation.

Individual-Based Evolution Control

The pseudo code of using individual-based evolution control to reproduce solutions for expensive evaluation is given in Algorithm 3. Two mating solutions are selected during this process, one is selected by distribution, another one is selected by convergence.

In order to maintain the diversity of non-dominated solutions S_{AF}, we employ reference vectors RV to divide the objective space into several regions S_R (line 1) [51] and count the number of non-dominated solutions in each region (line 2). The region containing the fewest non-dominated solutions is picked as the target region r_1^*(line 3), given it contains at least one non-dominated solution. Note that the regions without any non-dominated solution inside are ignored since there is no guarantee that non-dominated solutions exist in these regions. In r_1^*, non-dominated solutions with the largest relation value form the set $S_{r_1^*}$ (line 4), and based on normalised Euclidean distance, the sparsest solution in $S_{r_1^*}$ is selected as the first mating solution $\mathbf{x}^1$ (line 5).

For the purpose of convergence, the second mating solution $\mathbf{x}^2$ is randomly selected from reference points S_{RP} since all reference points are non-dominated solutions and they

Algorithm 3 Ind_Reproduce(h, RV, S_{AF}, S_V)

Input:

h: The ordinal-regression-based surrogate;
RV: Reference vectors in the m-dimensional objective space;
S_{AF}: Attainment front, the non-dominated solutions in archive S_A;
S_V: Quantified relation values.

Procedure:

1: $S_R = \{r_1, \ldots, r_{|RV|}\} \leftarrow$ Based on the Euclidean distance to RV, divide objective space into $|RV|$ regions.
2: Count the number of non-dominated solutions in each region.
3: $r_1^* \leftarrow$ The non-empty region with the fewest number of non-dominated solutions.
4: $S_{r_1^*} \leftarrow$ A set of non-dominated solutions with the largest relation value in region r_1^*.
5: $\mathbf{x}^1 \leftarrow$ Select the most sparse solution based on the density in $S_{r_1^*}$.
6: $r_2^* \leftarrow$ A random region with at least one reference point inside ($r_2^* \neq r_1^*$).
7: $\mathbf{x}^2 \leftarrow$ Select a random non-dominated solution from r_2^*.
8: Offspring population $P \leftarrow$ Crossover and mutation operations($\mathbf{x}^1, \mathbf{x}^2$).
9: $\mathbf{x}^* \leftarrow$ Select the best solution from P through Eq.(3.4).

Output:

$\mathbf{x}^*$: The candidate solution for expensive evaluation.

have the largest relation value. To avoid the situation where most of reference points are located in the same region, instead of selecting $\mathbf{x}^2$ from S_{RP} directly, a two-steps random selection is conducted. The first step selects a region r_2^* randomly, then $\mathbf{x}^2$ is selected from r_2^* (lines 6-7). A special case is where all solutions in S_{RP} are located in r_1^*, for this condition, a random solution $\mathbf{x}^2 \in S_{AF}$ but not in $r_1^*$is selected.

A crossover operator is applied to two mating solutions to produce two offspring solutions, then one randomly picked offspring solution is mutated to produce plenty of candidate solutions (line 8). These solutions will be evaluated on our ordinal-regression-based surrogate h and the best solution will be selected through Eq.(3.4).

3.3 Adaptive Ordinal Regression Evolutionary Algorithm (AOREA)

Using OREA as a basis, this section further proposes a novel SAEA to solve EMOPs with the ordinal-regression-based surrogate and an adaptive management strategy.

3.3.1 General Framework

The framework of our proposed adaptive ordinal regression evolutionary algorithm (AOREA) is depicted in Algorithm 4, which consists of four phases:

1. Initialisation: An initial dataset of size $11d$ - 1 is sampled from the decision space using the Latin hypercube sampling (LHS) [60] (line 1), where d is the dimensionality of decision variables. The sampled solutions are evaluated on the expensive problem f and then saved in an archive S_A (line 2). A group of reference vectors is generated using the method in [51] (line 3).

2. Ordinal relation quantification and surrogate modelling: The domination-based ordinal relations are quantified (line 5) and then used to train the ordinal-regression-based surrogate h (line 6).

3. Reproduction with adaptive sampling strategy: New solutions are generated using diversity maintenance, global search, or local search. The diversity maintenance method will be triggered once an imbalanced distribution of non-dominated solutions in S_A is detected (line 7, detailed in Algorithm 5). The states of recent optimisation are analyzed (line 8), which determines the strategy of evolutionary search in the current iteration (lines 9-15).

4. Update: New solutions are evaluated on the expensive functions (line 16), archive S_A is updated (line 17) and the number of used function evaluations FE is updated (line

Algorithm 4 AOREA framework

Input:

f: Objective functions of the expensive optimisation problem;
FE_{max}: Maximum number of evaluations allowed.

Procedure:

Sample a set of solutions $\{\mathbf{x}^1, \ldots, \mathbf{x}^{11d-1}\}$ and evaluate them on f.
Save all evaluated solution $(\mathbf{x}, f(\mathbf{x}))$ in an archive S_A. Set the number of used evaluations $FE = |S_A|$.
Generate reference vectors RV.
while $FE < FE_{max}$ **do**
  /*Ordinal Relation Quantification and Surrogate Modelling*/
  Quantify relation value v_i for all $\mathbf{x}^i \in S_A$, forming a set of relation values S_V.
  Train the ordinal surrogate $h \leftarrow \text{Kriging}(S_A, S_V)$
  /*Reproduction with Adaptive Management Strategy*/
  $\mathbf{x}^{2*}$, $n_{rep} \leftarrow$ Diversity Maintenance(h, RV, S_A, S_V).
  Update the state of optimisation search s.
  for $i = 1$ to n_{rep} **do**
    if random number $r < s$ **then**
      $\mathbf{x}^{i*} \leftarrow$ Global Search(h).
    else
      $\mathbf{x}^{i*} \leftarrow$ Local Search(h, S_A, S_V).
    end if
  end for
  /*Update*/
  Evaluate new solutions $\mathbf{x}^{1*}$ and $\mathbf{x}^{2*}$ on expensive functions f.
  Update $S_A = S_A \cup \{(\mathbf{x}^{1*}, f(\mathbf{x}^{1*})), (\mathbf{x}^{2*}, f(\mathbf{x}^{2*}))\}$.
  $FE = FE + 2$
end while

Output:

Non-dominated solutions in archive S_A.

18).

The ordinal-regression-based surrogate used in AOREA is the same as we discussed in Section 3.2.2. We are focusing on the adaptive management strategy in the next subsection as it is the major novelty of AOREA and helps us to distinguish AOREA from OREA.

3.3.2 Adaptive Management Strategy

The adaptive management strategy consists of two components: diversity maintenance and adaptive evolutionary search. The diversity maintenance will be triggered once an imbalanced distribution of non-dominated solutions is detected, while the adaptive evolutionary search will be conducted in every iteration.

Diversity Maintenance

The diversity maintenance method is derived from the individual-based evolution control in OREA. The distinction between the two methods is that the diversity maintenance method also considers the diversity when selecting the second mating solution. Besides, unlike the individual-based evolution control which will be conducted in every iteration of OREA, the diversity maintenance method will be triggered only when the solutions in S_{AF} are not well-distributed.

The reference vectors RV divide the objective space into several regions. In each iteration of optimisation, the distribution of non-dominated solutions S_{AF} in these regions is analyzed. As described in Algorithm 5, the number of non-dominated solutions in each region is counted (line 2). If the number of non-dominated solutions in a non-empty region is very few, or if the total number of non-dominated solutions is few (two criteria in line 6), then the diversity maintenance method will be triggered. In practice, two non-empty regions r_1^*, r_2^* with the fewest number of non-dominated solutions are selected as target regions (lines 3 and 7). To take both convergence and diversity into account, the two solutions with the largest relation values and the largest distance from other solutions are selected as mating parents (lines 8-9). Then crossover and mutation operations are conducted on mating parents to reproduce an offspring population P (line 10). Finally, an estimated optimal solution $\mathbf{x}^*$ is selected with the assistance of the ordinal-regression-based surrogate h (line 11). $\mathbf{x}^*$ is the solution we sampled from diversity maintenance for expensive evaluation. Otherwise, if the

Algorithm 5 Diversity Maintenance(h, RV, S_{AF}, S_V)

Input:

h: The ordinal-regression-based surrogate;
RV: Reference vectors in the m-dimensional objective space;
S_{AF}: Attainment front, the non-dominated solutions in archive S_A;
S_V: Quantified relation values.

Procedure:

$S_R = \{r_1, \ldots, r_{|RV|}\} \leftarrow$ Based on the Euclidean distance to RV, divide objective space into $|RV|$ regions.
Count the number of non-dominated solutions in each region.
$r_1^* \leftarrow$ The non-empty region with the fewest number of non-dominated solutions.
$n_{r_1^*} \leftarrow$ The number of non-dominated solutions in region r_1^*.
$n_{ne} \leftarrow$ The number of non-empty regions.
if $n_{r_1^*} < \frac{|S_{AF}|}{2n_{ne}}$ or $|S_{AF}| < 2m$ **then**
    $r_2^* \leftarrow$ Another non-empty region with the (second) fewest number of non-dominated solutions.
    Get the sets of non-dominated solutions with the largest relation values $S_{r_1^*}, S_{r_2^*}$.
    $\mathbf{x}^1, \mathbf{x}^2 \leftarrow$ Select solutions based on density in $S_{r_1^*}, S_{r_2^*}$, respectively.
    Offspring population $P \leftarrow$ Crossover and mutation operations($\mathbf{x}^1, \mathbf{x}^2$).
    $\mathbf{x}^* \leftarrow$ Select the best solution from P through Eq.(3.4).
    $n_{rep} = 1$.
else
    $\mathbf{x}^* \leftarrow$ NULL. /*No diversity maintenance in this generation.*/
    $n_{rep} = 2$.
end if

Output:

$\mathbf{x}^*$: The solution for expensive evaluation.
n_{rep}: Number of solutions to be reproduced in the adaptive evolutionary search.

criteria in line 6 are not satisfied, then we think the solutions in S_{AF} are well distributed already. As a result, there is no need to maintain diversity in this generation (line 14), two estimated optimal solutions will be generated by the adaptive evolutionary search (lines 9-15 in Algorithm 4).

Adaptive Evolutionary Search

The adaptive evolutionary search is derived from the generation-based evolution control in OREA. In adaptive evolutionary search, the output of either global search or local search is sampled for expensive evaluation. Note that both global and local search use the same evolutionary optimiser to search for optimal solutions, the optimiser can be any evolutionary optimisation method. In AOREA, the evolutionary optimiser used for global and local search is PSO [20], just like the setup we used in OREA. The evolutionary search aims to find the solution that maximises the expected improvement (described in Eq.(3.4)) using the surrogate h.

Difference Between Global and Local Search. The initial population in global search is uniformly sampled from the decision space, no prior is considered during the optimisation. However, priors are used to initialise the population in local search, which narrows down the range of evolutionary search in the decision space. Particularly, the mutants of the reference points S_{RP} in S_A are selected to initialise the population for local search.

State of Global Search. To determine which search method to use in the current iteration and to adapt the search method dynamically during the surrogate-assisted optimisation, we analyze the optimisation state of global search in the last few iterations. Particularly, we assume $\mathbf{x}^*$ is the solution reproduced by the last run of the global search, we set the state of

the global search in this run $s_1 = 1$ if $\mathbf{x}^*$ is non-dominated by the smooth AF:

$$\nexists \mathbf{x} \in S_{AF} : g_\lambda(\mathbf{x}) \prec g_\lambda(\mathbf{x}^\star) \tag{3.5}$$

where g_λ is a smooth function. $g_\lambda(\mathbf{x})$ is an m-dimensional vector with its i^{th} element $g_{\lambda,i}(\mathbf{x})$ defined as:

$$g_{\lambda,i}(\mathbf{x}) = \tilde{f}_i(\mathbf{x}) + \lambda \sum_{j=1}^{m} \tilde{f}_j(\mathbf{x}). \tag{3.6}$$

$\tilde{f}_i(\mathbf{x})$ is the i^{th} objective value of $\mathbf{x}$ that has been normalised by the range of m-dimensional objective space. λ is the factor of smoothness. Otherwise, the state $s_1 = 0$. It is noticeable that s_1 reflects whether the last run of global search has reached a satisfactory result or not.

Overall State of Optimisation. In the adaptive evolutionary search, states of the most recent k global searches are recorded. All states are initialised to 1 in the beginning. The overall state of optimisation in Algorithm 4 is updated using the mean of k states and a minimum state s_m:

$$s = max(\frac{\sum_{j=1}^{k} s_j}{k}, s_m). \tag{3.7}$$

The adaptive evolutionary search will run a global search if a random generated number $r \in [0, 1)$ is less than s, otherwise, a local search will be conducted. Therefore, if the global search cannot reach satisfactory results during a period of time in the surrogate-assisted optimisation, the probability of conducting global search will be decreased and the adaptive evolutionary search will allocate more effort to local search.

3.4 Experimental Studies

To evaluate the optimisation performance of the proposed OREA and AOREA on EMOPs, in this section, we conduct computational experiments to compare OREA and AOREA with

other SAEAs that are designed for EMOPs.

3.4.1 Experimental Setup

Optimisation Problems. The comparison experiments are conducted with the experimental setups that have been widely used in the domain of ECOP studies [68, 115, 90, 31]. Performance comparison is conducted on 7 DTLZ test problems [18], each problem instance contains 10 decision variables and 3 objectives. Latin hypercube sampling is used to generate 11 d - 1 data points as the initial dataset. The maximum number of allowed evaluations FE_{max} is set to 300. The statistical results are obtained from 30 independent runs.

Performance Indicator. The inverted generational distance plus (IGD+) [38] is employed as the performance indicator in our experiments. The IGD+ indicator measures the distance between the true Pareto front (reference points) and the attainment front, it takes both the diversity and the convergence of the attainment front into account. Given a set of reference points sampled from the Pareto front, the IGD+ indicator converts the inputted attainment front into a numerical value, making the results of multi-objective optimisation comparable. A small IGD+ value indicates that the inputted attainment front is close to the Pareto front, which means a good performance of a MOEA. In the experiments, the number of reference points is set to 5000, or a number very close to 5000, just as recommended in [68] [2]. To get reference points from the Pareto front, a series of evenly distributed reference vectors are generated, and each reference point $\mathbf{x}$ is the intersection of a vector and the true Pareto front of the problem to be optimised.

Comparison Algorithms. We compare OREA and AOREA with 6 state-of-the-art SAEAs designed for EMOPs, these SAEAs can be classified into three categories:

[2] Using more than 5000 reference points to compute IGD+ values will enhance the accuracy of performance estimation, but the improvement of accuracy is not significant and the time cost of computation will increase rapidly.

- Regression-based SAEAs: ParEGO [47], K-RVEA [8], and KTA2 [90]. ParEGO is a classic regression-based SAEA which employs a Kriging model to approximate an aggregate of all objectives. It uses expected improvement as infill sampling criterion and thus belongs to the family of Bayesian optimisation methods. K-REVA is a reference vector guided SAEA, it uses m surrogates to approximate m objectives. KTA2 is a newly proposed SAEA that is assisted by two archives.

- Classification-based SAEAs: CSEA [68] and REMO [31]. CSEA is a classic classification-based SAEA which employs a neural network to learn the binary relation between solutions and reference points. REMO is a newly proposed SAEA that learns the ternary relation between solutions with a neural network.

- Ordinal-regression-based SAEAs: OREA and AOREA. Both SAEAs use one ordinal-regression-based surrogate to approximate the domination-based ordinal landscape of all objectives.

The configuration of comparison algorithms and the code sources of them are same as [68]. Note that PRL is not compared here since it requires thousands of fitness evaluations [85]. Additionally, MOEA/D-EGO [120] and CPS-MOEA [118] are not compared in our experiments as they failed to outperform other comparison algorithms on any DTLZ test problem [31].

Parameter Setup. For the ordinal-regression-based surrogate in AOREA and OREA, the minimum number of ordinal levels used is $n_o = 10$, and the shaping coefficient $c_{shape} = 0.03$. The Kriging model is implemented using DACE [83] with the hyper-parameter range $\theta \in [10^{-5}, 10]$, $p \in [1, 2]$. In the diversity maintenance, the crossover and mutation operations are simulated binary crossover (SBX) [14] and polynomial mutation [15], respectively. The crossover probability $p_c = 1.0$ and crossover index $\eta_c = 20$; the mutation probability $p_m = 1/d$ and mutation index $\eta_m = 20$. The parameter setups of these reproductive operators

Table 3.1: Mean Inverted Generational Distance Plus (IGD+) values and standard deviation (in brackets) of 6 comparison algorithms and AOREA, results are obtained by 30 independent runs on DTLZ test problems with 10 variables and 3 objectives. '+', '≈', and '−' denote AOREA is statistically significantly superior to, almost equivalent to, and inferior to the compared algorithms in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. It can be observed that AOREA generally outperforms all comparison algorithms.

Problems	ParEGO	K-RVEA	KTA2	CSEA	REMO	OREA	AOREA
DTLZ1	5.98e+1(3.81e+0)+	8.88e+1(2.16e+1)+	4.75e+1(1.55e+1)+	6.30e+1(1.69e+1)+	5.06e+1(1.49e+1)+	4.44e+1(1.38e+1)+	3.83e+1(1.36e+1)
DTLZ2	2.61e-1(3.63e-2)+	9.22e-2(2.57e-2)+	3.82e-2(3.29e-3)−	1.60e-1(2.76e-2)+	1.01e-1(1.75e-2)+	5.86e-2(8.28e-3)≈	5.69e-2(7.00e-3)
DTLZ3	1.66e+2(1.31e+1)+	2.43e+2(4.61e+1)+	1.52e+2(4.73e+1)+	1.62e+2(4.84e+1)+	1.49e+2(3.88e+1)+	1.26e+2(3.18e+1)+	9.36e+1(2.82e+1)
DTLZ4	4.57e-1(7.52e-2)+	2.66e-1(1.02e-1)+	2.33e-1(8.36e-2)+	2.34e-1(7.76e-2)+	1.32e-1(6.41e-2)+	1.07e-1(9.68e-2)≈	7.04e-2(1.38e-2)
DTLZ5	1.60e-1(4.40e-2)+	9.18e-2(2.76e-2)+	8.66e-3(1.96e-3)−	9.58e-2(2.60e-2)+	5.78e-2(1.81e-2)+	1.59e-2(5.12e-3)≈	1.43e-2(3.30e-3)
DTLZ6	2.42e-1(1.07e-1)+	3.05e+0(5.23e-1)+	1.82e+0(4.48e-1)+	4.85e+0(6.38e-1)+	4.27e+0(5.48e-1)+	2.35e-1(4.14e-1)+	1.13e-1(1.76e-1)
DTLZ7	1.10e-1(3.57e-2)≈	7.39e-2(1.52e-2)−	1.54e-1(1.97e-1)≈	1.65e+0(6.43e-1)+	1.20e+0(5.73e-1)+	1.79e-1(1.20e-1)≈	2.00e-1(1.36e-1)
+/ ≈ /−	6/1/0	6/0/1	4/1/2	7/0/0	7/0/0	3/4/0	-/-/-

are suggested by the literature of comparison algorithms [120, 8, 68, 90]. The number of generated offspring solutions are $100d$. For the optimiser PSO in the adaptive evolutionary optimisation, the inertia coefficient is set to 0.5, and the learning rate of the cognitive component and that of the social component are both set to 1.5. The population size and neighbour size (with a ring topology) are set to 100 and 10, respectively. The maximum number of surrogate evaluations is 3000. States of the most recent $k = 10$ global search runs are recorded. The minimum state $s_m = 0.2$ and the factor of smoothness $\lambda = 0.05$. For local search, 20% solutions are initialised by conducting polynomial mutation operations on the reference points in S_{RP}.

3.4.2 Experimental Results

The statistical results of IGD+ values obtained by comparison algorithms are reported in Table 3.1. Wilcoxon rank sum tests are conducted to compare the results obtained by AOREA and comparison algorithms at a significance level of 0.05. It can be observed that AOREA has achieved the smallest IGD+ values on 4 DTLZ test problems, followed by OREA with the second smallest IGD+ values on 4 DTLZ test problems. The advantages of using ordinal-regression-based surrogates have been demonstrated. Besides, KTA2 has obtained the smallest IGD+ values on DTLZ2 and DTLZ5, K-RVEA has achieved the smallest

IGD+ value on DTLZ7. In general, AOREA outperforms most state-of-the-art comparison algorithms on DTLZ test problems.

The comparison between AOREA and OREA shows that the use of the adaptive management strategy improves the multi-objective optimisation performance of OREA on DTLZ test problems. Clearly, on DTLZ1, DTLZ3, and DTLZ6, statistically significant improvements are achieved. This is explainable as OREA uses a fixed management strategy in which all optimisation resources are allocated to global search and diversity maintenance evenly, thus no special effort has been made to do a local search. However, DTLZ1 and DTLZ3 are multimodal problems with rugged landscapes. It is difficult for SAEAs to converge within limited evaluations, especially for the SAEAs that mainly use global optimisers. In addition, the optimum of DTLZ6 is located in the corner of the decision space. Although global search plays an important role in the early stage of optimisation, local search is more desirable in the late stage. In contrast, AOREA takes the advantages of the adaptive management strategy and thus expends some effort on local search. Therefore, significant improvement can be observed when compared with OREA. A different example is DTLZ4, which is a scaled problem with poor solution distribution in the objective space. Global search is preferable to local search on DTLZ4. AOREA and OREA have achieved comparable results on DTLZ4. This indicates the adaptive management strategy has adapted itself dynamically during the optimisation, rather than simply sampling more solutions from local search. Additionally, the dynamic diversity maintenance in the adaptive management strategy also makes the allocation of optimisation resources in AOREA more flexible than OREA. This might explain why AOREA always outperforms OREA on DTLZ test problems.

The distribution of non-dominated solutions obtained by AOREA on 4 DTLZ test problems is illustrated in Fig. 3.2. Although AOREA generally outperforms the compared optimisation algorithms, the non-dominated solutions found by AOREA are not well-distributed on DTLZ4 and DTLZ7. Particularly, on DTLZ7, a patch of the true Pareto front is missing, which leads to a worse IGD+ performance than K-RVEA and ParEGO in

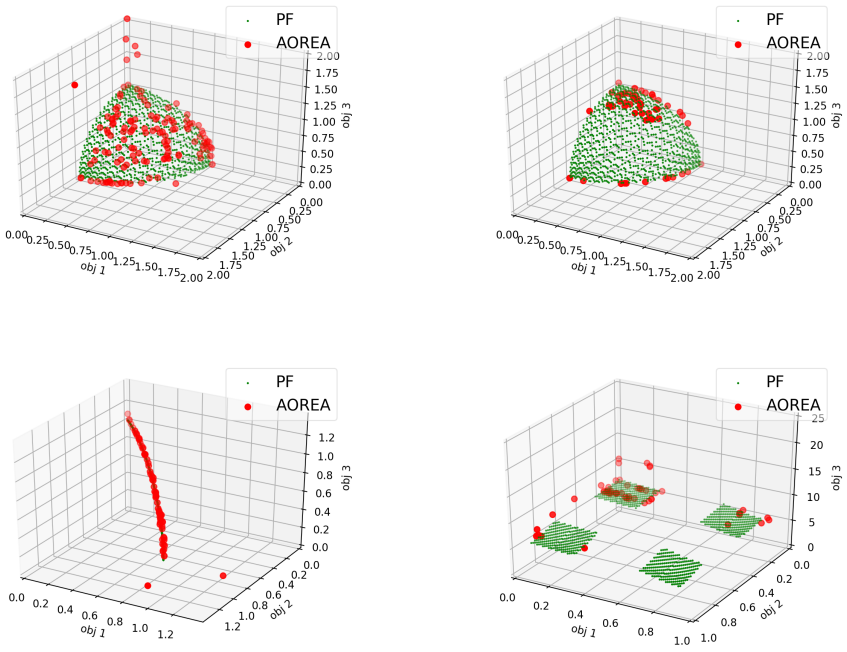


Figure 3.2: The distribution of non-dominated solutions obtained by AOREA on DTLZ2 (upper left), DTLZ4 (upper right), DTLZ5 (lower left), and DTLZ7 (lower right) in the run associated with the median IGD+ value.

Table 3.1. This is caused by the limitation of the ordinal-regression-based surrogate. The ordinal-regression-based surrogate cannot predict the location of solutions in the objective space, which lowers the performance of our diversity maintenance method. Similar limitations exist in other classification-based SAEAs. By comparison, K-RVEA and ParEGO are regression-based SAEAs that supported by reference vectors, thus they are suitable for the problem with discontinuous Pareto front.

3.5 Chapter Summary

In this chapter, we present two effective SAEAs for solving EMOPs, namely OREA and AOREA. Both SAEAs use a novel ordinal-regression-based Kriging surrogate as their surrogates. Such a surrogate is designed to approximate the domination-based ordinal landscape for all objectives and thus is capable of learning the ordinal relations between solutions ef-

ficiently. On the basis of this ordinal surrogate, a hybrid surrogate management strategy and an adaptive surrogate management strategy are proposed to save expensive function evaluations with computational efficient surrogate evaluations. The proposed adaptive management strategy consists of diversity maintenance and adaptive evolutionary search. In each iteration, the diversity maintenance cooperates with the reference vectors to analyze the distribution of non-dominated solutions. More specifically, based on the regions of the objective space divided by reference vectors, the diversity maintenance method checks if a non-empty region has much fewer non-dominated solutions than other regions. Such an imbalanced distribution of non-dominated solutions will trigger a reproduction procedure, which generates new solutions to improve the diversity and balance the distribution. Moreover, a small total number of non-dominated solutions will also activate the diversity maintenance method. The adaptive evolutionary search allocates the optimisation resources to either global search or local search. During the optimisation, the optimal solution found by global search will be compared with the non-dominated solutions in the archive. The comparison result will affect the state of global search, which determines the probability of conducting global search in the upcoming iterations. Besides, in an iteration, when the diversity maintenance has not been activated, more optimisation resources will be allocated to global or local search.

Our experimental studies on DTLZ test problems have demonstrated that OREA and AOREA generally outperform all compared state-of-the-art SAEAs. Particularly, the advantages and the effectiveness of the adaptive management strategy have been demonstrated in the comparison between OREA and AOREA. It turns out that the dynamic adaptation between global search, local search, and diversity maintenance is able to improve the optimisation performance of SAEAs designed for EMOPs.

The contributions of this chapter can be summarised as:

- An ordinal-regression-based surrogate is proposed. Different from existing surrogates which are employed to either approximate accurate fitness values (regression-based

surrogates) or learn binary/ternary solution relations (classification-based surrogates), our ordinal surrogate manages to approximate the domination-based ordinal landscape of all objectives. The approximation of ordinal landscape is a more efficient representation of solution relations than binary or ternary classifiers. Moreover, the landscape of all objectives is approximated with one ordinal surrogate instead of a combination of multiple regression-based surrogates, which indicates the computational efficiency of using our ordinal surrogate in SAEAs. This is an answer to the first question in Section 1.3.1.

- A novel SAEA (Ordinal Regression Evolutionary Algorithm, OREA) is developed to solve ECOPs with our ordinal-regression-based surrogate. To manage our ordinal surrogate in OREA, a hybrid surrogate management strategy is developed, which is derived from the generation-based evolution control framework and the individual-based evolution control framework. Such a hybrid strategy determines how to generate candidate solutions and which solutions will be selected for re-evaluation on expensive functions. This is an answer to the second question in Section 1.3.1.

- An adaptive management strategy is proposed to manage surrogates for SAEAs designed for EMOPs. Based on the state of optimisation in the last few iterations, the proposed strategy is able to dynamically balance global search and local search. Moreover, a dynamic diversity maintenance method will be triggered once the imbalanced distribution of non-dominated solutions is detected. The proposed adaptive management strategy is introduced to OREA as a surrogate management strategy, resulting in a new algorithm, adaptive ordinal regression evolutionary algorithm (AOREA). This is an answer to the second question in Section 1.3.1.

Chapter 4

Surrogate-Assisted Bilevel Evolutionary Algorithm for Expensive Constrained Optimisation

The previous chapter aimed to address unconstrained EMOPs by using ordinal-regression-based SAEAs. Considering constrained optimisation is a common optimisation scenario in the domain of expensive optimisation and it can be encountered in many real-world applications [116]. In this chapter, we focus on a real-world gasoline engine calibration problem. Specifically, our research on ECOPs is conducted on the basis of a real-world gasoline engine calibration problem. The problem has six engine control parameters including throttle angle, waste gate orifice, ignition timing, valve timings, state of injection, and air-fuel-ratio. The problem aims at minimising the brake special fuel consumption (BSFC) and satisfying four constraints in terms of temperature, pressure, CA50, and load simultaneously. Due to the limited financial budget and short calibration cycle, only 300-500 evaluations are available during the calibration process. Hence, this engine calibration problem can be

formulated as an expensive constrained optimisation problem (ECOP):

$$\begin{aligned} &\text{minimise} \quad f(\mathbf{x}) \\ &\text{subject to} \quad 0 \leq g_i(\mathbf{x}) \leq 1, \quad i = 1, 2, 3, 4, \end{aligned} \tag{4.1}$$

where $\mathbf{x}$ is a solution of six engine control parameters, f is the expensive objective to be minimised, g_i are constraints. This chapter proposes a surrogate-assisted bilevel evolutionary algorithm to solve a real-world engine calibration problem. Principal component analysis is performed to investigate the impact of variables on constraints and to divide decision variables into lower-level and upper-level variables. The lower-level aims at optimising lower-level variables to make candidate solutions feasible, and the upper-level focuses on adjusting upper-level variables to optimise the objective. In addition, an ordinal-regression-based surrogate is adapted to estimate the ordinal landscape of solution feasibility. Computational studies on a gasoline engine model demonstrate that our algorithm is efficient in constraint handling and also achieves a smaller fuel consumption value than other state-of-the-art calibration methods.

4.1 Motivation

SAEAs have been widely used to solve ECOPs [104, 106, 121]. These SAEAs employ surrogates to approximate the landscape of objectives and constraints and predict the solution feasibility of candidate solutions, which saves valuable evaluations. Efficient constraint handling techniques (CHTs) are desirable for SAEAs to explore the feasible region. As discussed in Section 2.3.2, existing CHTs tend to allocate the optimisation resource to every decision variable evenly. Interestingly, in some engine calibration problems, the distribution of local optima implies that some decision variables are playing more important roles than other variables [97]. Therefore, the feasibility of solutions could be mainly affected by only a subset of decision variables, which motivates us to handle constraints with the architecture of

bilevel optimisation.

We use a brief introduction to bilevel optimisation to inspire the bilevel architecture we used in this chapter. However, the ECOP considered in this chapter is not a bilevel optimisation problem. We are not considering bilevel optimisation in this chapter. Bilevel optimisation is defined as a mathematical program, where an upper-level optimisation problem contains another lower-level optimisation problem as a constraint [89]. Clearly, an upper-level solution is feasible only if it is one of the optima of a lower-level optimisation problem [17]. A simplified bilevel optimisation problem can be formulated as follows:

$$\begin{aligned} &\text{minimise} \quad f_u(\mathbf{x}_u, \mathbf{x}_l) \\ &\text{subject to} \quad \underset{\mathbf{x}_l}{\text{argmin}} \quad f_l(\mathbf{x}_u, \mathbf{x}_l), \end{aligned} \tag{4.2}$$

where f_u and f_l are the objective of upper-level and lower-level optimisation, respectively. $\mathbf{x}_u$ is a vector of upper-level variables, $\mathbf{x}_l$ is a vector of lower-level variables. Note that lower-level optimisation function f_l depends on both $\mathbf{x}_u$ and $\mathbf{x}_l$ instead of only $\mathbf{x}_l$. Further details on bilevel optimisation and its architecture are available in [89]. To solve our engine calibration problem with such a bilevel architecture, we can set the objective to be minimised f as upper-level objective f_u and set four constraints g as lower-level objectives f_l. Note that in this bilevel architecture, the membership of $\mathbf{x}_u$ and $\mathbf{x}_l$ is unknown and needs to be identified. Since the lower-level optimisation is focusing on lower-level variables, if the decision variables that have significant impacts on solution feasibility can be identified as lower-level variables, then more optimisation effort will be made on them. Consequently, the efficiency of searching for the feasible region will be enhanced when compared with other CHTs.

This chapter analyzes the sensitivity of solution feasibility to decision variables, the decision variables that have a significant impact on solution feasibility will be divided into lower-level variables. A bilevel architecture is employed to handle constraints where more

efforts are made on lower-level variables to explore the feasible region. Besides, to improve the efficiency of surrogate modelling, an ordinal-regression-based surrogate will be adapted to approximate constraints.

4.2 Surrogate-Assisted Bilevel Differential Evolution (SAB-DE)

4.2.1 General Framework

A diagram of the proposed SAB-DE framework is shown in Fig. 4.1. The pseudo code of

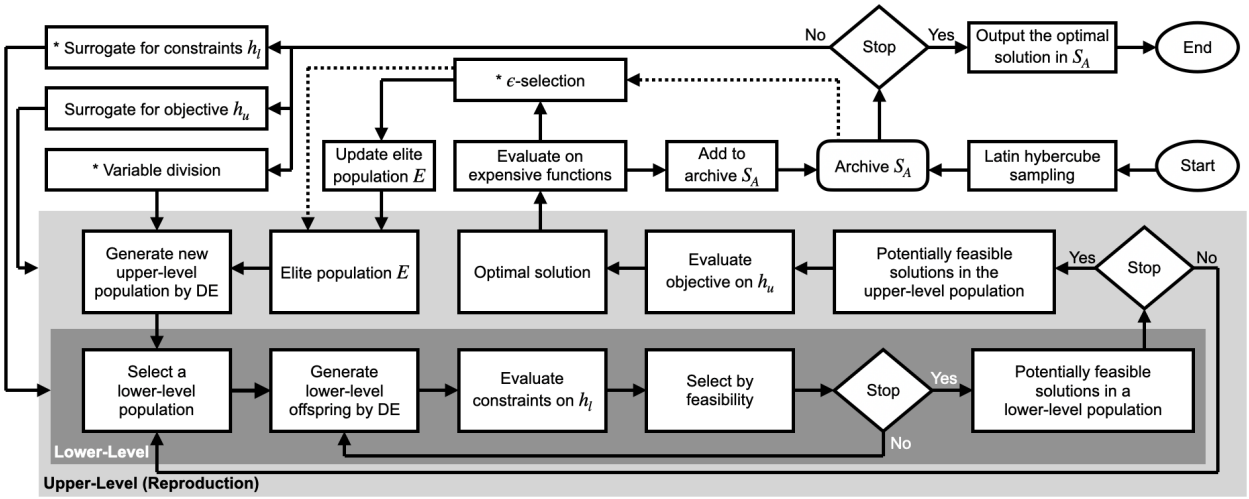


Figure 4.1: Diagram of the proposed SAB-DE framework. It can be seen that the bilevel architecture is nested rather than parallel: The lower-level optimisation (in the grey block) works as a component of the reproduction operation of the upper-level optimisation (in the light grey block). The dash lines will be executed only once for the purpose of initialisation. Note that the modules marked by star ($*$) symbols as well as the architecture of bilevel optimisation for ECOPs are our major contributions.

the proposed SAB-DE framework is depicted in Algorithm 6. The algorithm consists of the following steps.

1. Initialisation: N_{init} solutions are sampled from the decision space using the Latin hypercube sampling (LHS) method [60] (line 1). These solutions are evaluated on expensive

Algorithm 6 Framework of SAB-DE

Input:

f, g: Objective function, constraint functions;
N_{init}: Size of the initial archive S_A;
FE_{max}: Maximum number of allowed evaluations;
N_e: Size of the elite population E;
N_u: Size of the upper-level population U;
N_l: Size of the lower-level population L.

Procedure:

1: $D_x \leftarrow$ Latin_Hypercube_Sampling (N_{init})
2: $S_A = \{D_x, f(D_x), g(D_x)\}$
3: $FE = N_{init}$
4: **while** $FE < FE_{max}$ **do**
5: $X_l, X_u \leftarrow$ Variables_Division(S_A)
6: $h_l, h_u \leftarrow S_A$ /*Surrogates.*/
7: **if** $FE == N_{init}$ **then**
8: Initialise $E \leftarrow \epsilon$-Selection $(f, S_A, h_l, \emptyset, N_e, NULL)$
9: **else**
10: Update $E \leftarrow \epsilon$-Selection $(f, S_A, h_l, E, N_e, \mathbf{x}^*)$
11: **end if**
12: Optimal solution $\mathbf{x}^* \leftarrow$Reproduction$(E, S_A, N_u, N_l, X_u, X_l, h_u, h_l)$
13: Evaluate $\mathbf{x}^*$ on f, g
14: $FE = FE + 1$
15: Update $S_A = S_A \cup \{\mathbf{x}^*, f(\mathbf{x}^*), g(\mathbf{x}^*)\}$
16: **end while**

Output:

Best feasible solution in S_A.

functions f, g and then saved in an archive S_A (line 2). Note that LHS is a statistical sampling method that maximise the diversity of sampling different combinations of different variable values. Using LHS to initialise S_A enhances the initial diversity of S_A.

2. Decision Variable Division: All decision variables X are divided into either upper-level variables X_u or lower-level variables X_l (line 5). The division of upper and lower variables is based on the quantified variable impacts, which are estimated by a PCA-based sensitivity analysis. No prior knowledge is required in this step.

3. Surrogates Training: The surrogates in both upper-level and lower-level are trained with S_A (line 6). The surrogate in upper-level optimisation h_u approximates the fitness landscape of the objective function f. The surrogate in lower-level optimisation h_l approximates the ordinal landscape of solution feasibility.

4. Environmental Selection: N_e solutions in S_A are selected to initialise an elite population E using an ϵ-selection strategy (lines 7-8). Otherwise, the newly evaluated solution $\mathbf{x}^*$ from the last iteration is used to update E (lines 9-11).

5. Reproduction: An upper-level population U of size N_u is generated by conducting DE operations on elite population E and archive S_A. n_s upper-level variable vectors $\mathbf{x}_u$ are produced, each $\mathbf{x}_u^i$ corresponds to a subpopulation of size N_l (denoted by lower-level population L_i). Then, for each L_i, an independent lower-level optimisation (step 6) is conducted. Once all potentially feasible solutions have been collected from the lower-level optimisation, an unconstrained optimisation will be conducted to produce an estimated optimal solution $\mathbf{x}^*$. The structures of U and L are illustrated in Fig. 4.2.

6. Lower-Level Optimisation: The lower-level optimises lower-level variables X_l to generate more potentially feasible solutions. This step is a component of step 5.

7. Evaluation and Update Archive: Evaluate the output of the upper-level optimisation $\mathbf{x}^*$ on expensive functions f, g (line 13). Update archive S_A (line 15).

8. Output the optimal feasible solution in S_A when evaluation budget FE_{max} has run out.

In the following subsections, we first introduce the method to divide decision variables (step 2). Then, we describe the surrogates in both upper-level and lower-level optimisation (step 3). Finally, we discuss the details of the upper-level optimisation, including the ϵ-

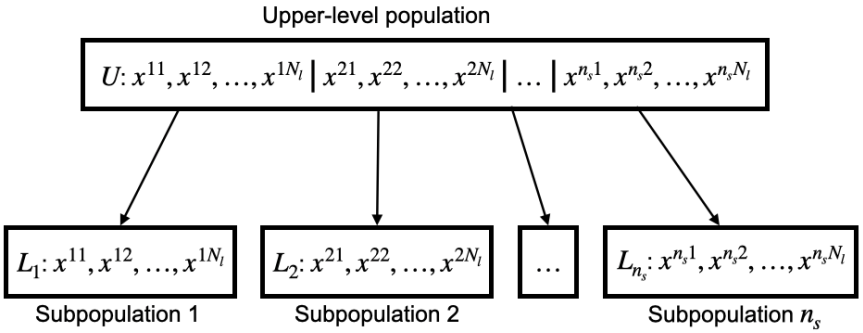


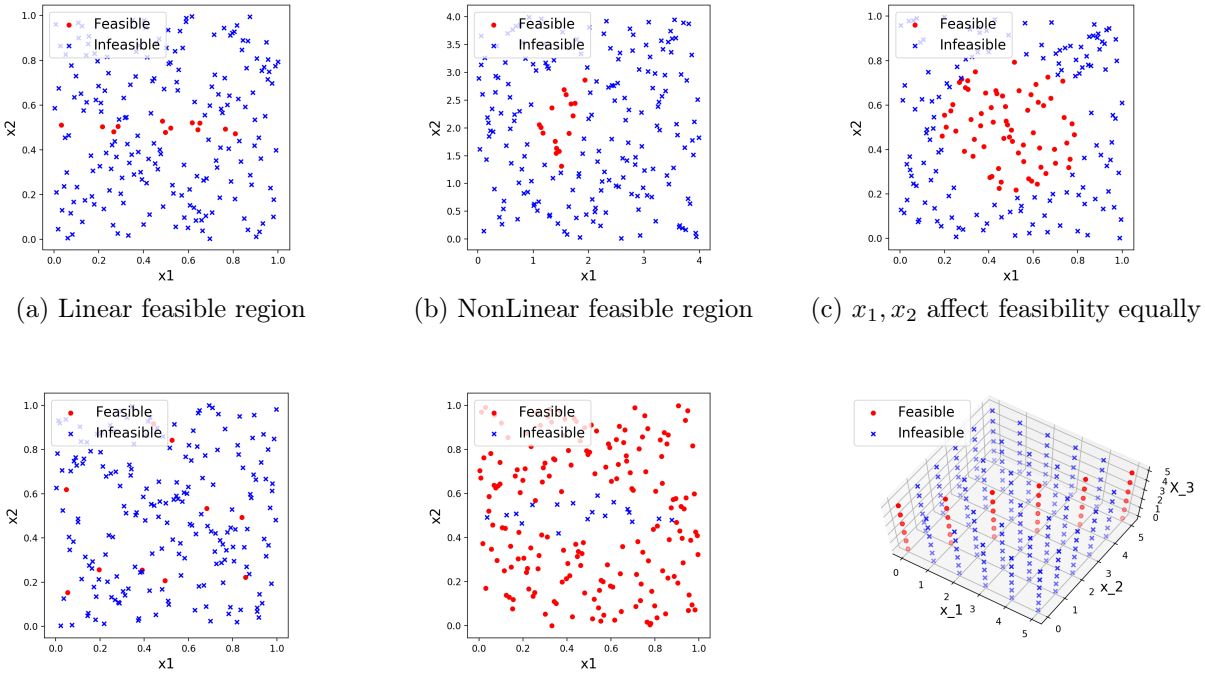
Figure 4.2: Illustration of the structures of the upper-level population U and lower-level populations L_i, $i = 1, \ldots, n_s$. There are N_u solutions in U, each solution $\mathbf{x}^{ij}$ is a combination of an upper-level variable vector $\mathbf{x}_u^i$ and a lower-level variable vector $\mathbf{x}_l^{ij}$, presented in the format: $\mathbf{x}^{ij} = (\mathbf{x}_u^i, \mathbf{x}_l^{ij})$. Solutions in U can be evenly divided into n_s groups. Each group contains $N_l = N_u/n_s$ solutions, forming a subpopulation (denoted by lower-level population L_i). For a given L_i, all solutions $\mathbf{x}^{ij}$ in L_i share the same $\mathbf{x}_u^i$, thus U has n_s different upper-level variable vectors $\mathbf{x}_u^i$ in total.

selection strategy developed to maintain the elite population E (step 4), the reproduction operation of the upper-level optimisation (step 5), and the lower-level optimisation (step 6).

4.2.2 Division of Upper-Level and Lower-Level Variables

The variable division aims at allocating the decision variables that have greater impacts on solution feasibility than other variables to lower-level variables X_l. Thus, we need to quantify the variable impacts before the variable division. In the decision space, if an independent variable x_i is essential to solution feasibility, then solution feasibility will vary in the corresponding i^{th} dimension, implying the distribution of x_i of feasible solutions or infeasible solutions will be limited in a relatively narrow range. Fig. 4.3 illustrates how the variables x_i, which have greater impacts on solution feasibility, affect the distribution of feasible (red circle) or infeasible (blue cross) solutions in the decision space. Hence, the basic idea behind the quantification of variable impacts is to investigate the distribution of feasible solutions. Note that the investigation of the distribution of infeasible solutions is ignored in our variable division process. Although the distribution of infeasible solutions can be more informative than that of feasible solutions in the case presented in Fig. 4.3e, the feasible region will be

much larger than the infeasible region. In such a situation, it is not necessary to use special constraint handling methods to search for feasible solutions.



(a) Linear feasible region (b) NonLinear feasible region (c) x_1, x_2 affect feasibility equally

(d) x_1, x_2 affect feasibility equally (e) Linear infeasible region (f) x_3 has no impact on feasibility

Figure 4.3: Distribution of feasible solutions in the decision space. The feasible region in Fig. (a) is shaped by a linear constraint. Feasible solutions are scattered over a narrow range in the dimension x_2, because x_2 has a significant impact on solution feasibility. The feasible region in Fig. (b) is shaped by two nonlinear constraints. x_1 has a greater impact on solution feasibility than x_2. All variables in Figs. (c) and (d) have equal impacts on solution feasibility. In Fig. (e), the distribution of infeasible solutions implies that x_2 affects the feasibility of infeasible solutions heavily. Fig. (f) displays a 3D decision space, where x_1 and x_2 have equal impacts on solution feasibility but x_3 is irrelevant to solution feasibility.

We use variance to measure whether a decision variable x_i is scattered over a narrow range or not. The interaction between decision variables should be considered. For example, in Fig. 4.3f, three original variables x_1, x_2, x_3 of feasible solutions have equal variance. But it is obvious that x_1 and x_2 have equal impacts on solution feasibility but x_3 is irrelevant to that. To take the variable interaction into account, we employ PCA to identify the principal components from original variables, and then measure the variance of these principal components. The pseudo code of decision variable division is presented in Algorithm 7. First, PCA

Algorithm 7 Variable_Division

Input:

X: d-dimensional space of decision variables;
S_A: Archive of evaluated solutions.

Procedure:

1: $S_f \leftarrow S_A$ /*Feasible solutions in S_A.*/
2: **if** $|S_f| < 2$ **then**
3: $X_l \leftarrow$ Select $\lfloor \frac{d}{2} \rfloor$ variables from X randomly.
4: **else**
5: $n_c = min\{|S_f|, d\}$
6: $\mathbf{vr}, \mathbf{c} \leftarrow PCA(S_f)$ /*Variance ratio **vr**,components **c**.*/
7: $\mathbf{vr} = \frac{1}{0.001+\mathbf{vr}}$
8: **for** $i = 1$ to d **do**
9: $imp_i = \sum_{j=1}^{n_c} vr_j c_{ji}^2$ /*Quantified variable impacts.*/
10: **end for**
11: $\mathbf{P}_{X_l} = \frac{\mathbf{imp}}{\mathbf{imp}+d}$ /*Probability of X_l.*/
12: Allocating variables to X_l with probability vector $\mathbf{P}_{X_l}$.
13: **end if**
14: $X_u = X - X_l$ /*Set operation.*/

Output:

X_u: Set of upper-level variables;
X_l: Set of lower-level variables.

is performed on feasible solutions S_f to obtain the principal components **c** and the ratio of variance explained by these components **vr** (line 6). Considering a small variance indicates a significant impact on solution feasibility, the variance ratio **vr** is inverted and then used as an indicator of component impact on solution feasibility. A constant 0.001 is added to the denominator to smooth such a component impact indicator (line 7). Then, we quantify the impact of each decision variable x_i on solution feasibility as:

$$imp_i = \sum_{j=1}^{n_c} vr_j c_{ji}^2, \tag{4.3}$$

where n_c is the number of principal components, r_j is the j^{th} element of $\mathbf{r}$. c_{ji} is the i^{th} element of the j^{th} principal component c_j, which presents the contribution of x_i to c_j (lines 8-10). The quantified impact imp_i is a cumulation of inverted variance ratios of x_i on all

principal components.

To determine which variables are lower-level variables, the quantified variable impacts **imp** are not used directly. In practice, we use the ratio of **imp** to the sum of **imp** and d as the probability $\mathbf{P}_{X_l}$ of allocating decision variables to X_l (line 11). $\mathbf{P}_{X_l} \in (0,1)^d$ and a large imp_i indicates variable x_i has a great probability to be allocated to X_l. There are two reasons to introduce probability as uncertainty in the variable division: Firstly, the archive S_A is a small dataset sampled from the real distribution, so the impact quantified from S_A can be different from the real one, leading to an incorrect variable division. If the division does not contain uncertainty, the incorrect division has no chance to be corrected. Secondly, although most optimisation resource in the lower-level optimisation is allocated to the variables which have significant impacts on solution feasibility, other variables can still get small chances to be optimised due to their small impacts on solution feasibility. Noted that X_u and X_l cannot be $\emptyset$, otherwise, the variable x_i with minimum or maximum probability $\mathbf{P}_{X_l}$ will be allocated to X_u or X_l, respectively.

The effectiveness of the PCA operation in the variable division process is demonstrated using the example illustrated in Fig. 4.3f. For three variables, $\mathbf{P}_{X_l}$=$\{0.994, 0.994, 0.5\}$. Hence, x_1, x_2 have a probability of 99.4% to be allocated to X_l. In comparison, if we use the variance of decision variables directly without a PCA operation, all elements in $\mathbf{P}_{X_l}$ will be 0.5, implying all variables have a probability of 50% to be allocated to X_l. The impacts of x_1 and x_2 on solution feasibility have not been quantified correctly. Therefore, PCA operation is necessary for the process of sensitivity analysis. The sensitivity analysis is based on S_A. Thus, it does not require extra sampling effort and it is not computational costly.

4.2.3 Surrogates in Bilevel Architecture

In this work, we use Kriging models [91] as our surrogates for both upper-level and lower-level optimisation. The engine calibration problem to be solved has only one objective, so

the upper-level optimisation employs a Kriging model as its surrogate to approximate the fitness. Instead of training four separate surrogates to approximate constraints, we adapt a Kriging model as an ordinal-regression-based surrogate [115] in the lower-level optimisation. The use of our ordinal surrogate implies the lower-level optimisation needs to maintain only one surrogate rather than four separate surrogates. Therefore, the architecture of the lower-level optimisation is simplified and the efficiency of surrogate management is improved. All decision variables are used to train our upper-level and lower-level surrogates.

Regression-Based Surrogate for the Objective

The Kriging surrogate in the upper-level optimisation h_u aims to approximate the fitness, which is the most common usage of the Kriging model in optimisation [46, 109]. The correlation function of h_u is defined as:

$$Corr(\epsilon(\mathbf{x}^i), \epsilon(\mathbf{x}^j)) = exp[-dis(\mathbf{x}^i, \mathbf{x}^j)], \tag{4.4}$$

where the distance between two solutions $\mathbf{x}^i$ and $\mathbf{x}^j$ is:

$$dis(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=1}^{d} \theta_k |x_k^i - x_k^j|^{p_k}. \tag{4.5}$$

$\boldsymbol{\theta}$ and $\mathbf{p}$ are two vectors of parameters that are tuned by maximising the following likelihood:

$$\frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|\mathbf{R}|^{1/2}} exp[-\frac{(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}], \tag{4.6}$$

where $|\mathbf{R}|$ is the determinant of the correlation matrix, μ, σ^2 are the mean and the variance of the prior distribution, respectively. $\mathbf{y}$ is the vector of fitness in S_A.

For a given candidate solution $\mathbf{x}^*$, the Kriging surrogate h_u produces a predictive Gaussian distribution $\mathcal{N}(\hat{y}(\mathbf{x}^*), \hat{s}^2(\mathbf{x}^*))$, the predicted mean $\hat{y}(\mathbf{x}^*)$ and covariance $\hat{s}^2(\mathbf{x}^*)$ are

specified as [46]:

$$\hat{y}(\mathbf{x}^*) = \mu + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu), \tag{4.7}$$

$$\hat{s}^2(\mathbf{x}^*) = \sigma^2(1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r}), \tag{4.8}$$

where $\mathbf{r}$ is a correlation vector consisting of covariances between $\mathbf{x}^*$ and S_A, other variables are explained in Eq.(4.6). The selection criterion of h_u is expected improvement (EI), as suggested in [46]. More details about the Kriging model are available in Chapter 2.

Ordinal-Regression-Based Surrogate for Multiple Constraints

The ordinal-regression-based Kriging surrogate h_l in the lower-level optimisation is trained in the same way as the upper-level surrogate h_u except the fitness vector $\mathbf{y}$ in Eq.(4.6) is replaced by a relation value vector. Therefore, in this subsection, we focus on the quantification of domination-based ordinal relations for constraints.

The relation value is a numerical label that is designed to quantify the domination-based ordinal relation between a solution and the reference points in the objective space [115]. Considering the lower-level optimisation is handling constraints instead of objectives, the reference points in the ordinal-regression-based surrogate are replaced by the real constraints in our engine calibration problem. Four constraints in Eq.(4.1) are redefined as:

$$\begin{aligned} & g_i^*(\mathbf{x}) = abs(g_i(\mathbf{x}) - 0.5) \\ & \textit{subject to} \quad g_i^*(\mathbf{x}) \leq 0.5, \quad i = 1, 2, 3, 4. \end{aligned} \tag{4.9}$$

In this way, the left-side constraints in Eq.(4.1) can be ignored since $g_i^*(\mathbf{x})$ is an absolute value that is always non-negative. Thus we define the reference points as 4 unit vectors $\{e_1, e_2, e_3, e_4\}$. For i^{th} reference point e_i, its i^{th} entry is 0.5 and the remaining entries are 0.

An extension coefficient ec is calculated beforehand using the domination-based ordinal relation between the reference points and a solution $\mathbf{x}$ in S_A, which can be simplified in

this problem:

$$ec(\mathbf{x}) = \max_{\text{over } i} \left(\frac{g_i^*(\mathbf{x})}{0.5} \right), \quad i = 1, 2, 3, 4. \tag{4.10}$$

Multiplying reference points by the extension coefficient $ec(\mathbf{x})$ will result in extended constraints. The solution $\mathbf{x}$ will be non-dominated with respect to the extended constraints in the constraint space. Therefore, ec is a measure of the distance between a solution $\mathbf{x}$ and the constraint boundaries. The solutions in S_A are sorted based on their ec values and then divided into N_o ordinal values:

$$N_o = max(n_o, |S_A|/|S_f|), \tag{4.11}$$

where n_o is the minimum number of ordinal levels defined by users, S_f is the set of feasible solutions. Clearly, all feasible solutions S_f are allocated to the first ordinal level, and remaining infeasible solutions are uniformly divided into $N_o - 1$ ordinal levels based on their rankings in ec. Finally, for solutions in the i^{th} ordinal level, their relation value $v_i = \frac{i-1}{N_o-1}$. These relation values are used to train the lower-level surrogate h_l.

As for the usage of surrogate h_l, a given candidate solution $\mathbf{x}$ is predicted to be feasible if $h_l(\mathbf{x}) \leq \frac{1}{2(N_o-1)}$ or infeasible if $h_l(\mathbf{x}) > \frac{1}{2(N_o-1)}$. The feasibility of two solutions $\mathbf{x}^i$ and $\mathbf{x}^j$ can be compared even if they are both infeasible, since the relation values $h_l(\mathbf{x}^i)$ and $h_l(\mathbf{x}^j)$ are numerical and thus comparable.

4.2.4 Upper-Level Optimisation

In SAB-DE, the upper-level optimisation focuses on optimising the objective, while the lower-level optimisation concentrates on constraint handling. The upper-level population U is generated from elite population E. To exploit the infeasible solutions that are close to the feasible region, an ϵ-selection strategy is developed to initialise E from S_A and to update E iteratively during the optimisation.

Algorithm 8 ϵ-Selection

Input:

f: Objective function;
S_A: An archive of evaluated solutions;
h_l: Lower-level surrogate;
E: Elite population;
N_e: Size of the elite population;
$\mathbf{x}^*$: The solution to be selected.

Procedure:

1: $S_A \leftarrow$ Sort S_A by lower-level ordinal labels $h_l(S_A)$.
2: $S_C \leftarrow$ top-ϵ infeasible solutions in S_A. /*Set of candidate solutions that are close to the feasible region.*/
3: $S_C \leftarrow S_C \cup \{$All feasible solutions in $S_A\}$.
4: **if** E is $\emptyset$ **then**
5: $S_C \leftarrow$ Sort S_C by their objective values $f(S_C)$.
6: $E \leftarrow$ Optimal N_e solutions in S_C. /*Initialise E.*/
7: **else**
8: $\mathbf{x}^k \leftarrow$ A solution randomly picked from E.
9: **if** $\mathbf{x}^* \in S_C$ and $f(\mathbf{x}^*) < f(\mathbf{x}^k)$ **then**
10: The k^{th} solution in $E \leftarrow \mathbf{x}^*$. /*Update E.*/
11: **end if**
12: **end if**

Output:

E: Initialised or updated E.

Environmental Selection by $\epsilon-$Selection Strategy

The ϵ-selection strategy is outlined in Algorithm 8. In the ϵ-selection strategy, the parameter $\epsilon \in [0, 1]$ is a criterion for determining if an infeasible solution is close to the feasible region or not. For solutions that have been sorted by their distance to the feasible region (line 1), only feasible solutions and top-ϵ infeasible solutions will be selected as candidate solutions S_C (line 2-3). For example, assuming S_A has 65 solutions, including 5 feasible and 60 infeasible solutions. If $\epsilon = \frac{1}{3}$, then $60 \times \frac{1}{3} = 20$ infeasible solutions as well as 5 feasible solutions will be selected and included in S_C, the size of S_C will be 25. The selected solutions in S_C will be further sorted by objective values (line 5), and the optimal N_e solutions in S_C are selected to initialise E (line 6).

The solution evaluated in the last iteration $\mathbf{x}^*$ will be used to update a randomly picked solution $\mathbf{x}^k$ from E, where $k \in \{1, \ldots, N_e\}$ (line 8). Note that $\mathbf{x}^*$ is used to update a randomly picked $\mathbf{x}^k$, instead of the worst one, from E in order to enhance population diversity and avoid premature convergence. If $\mathbf{x}^* \in S_C$ and the fitness of $\mathbf{x}^*$ is smaller than that of $\mathbf{x}^k$ (line 9), then $\mathbf{x}^*$ will replace $\mathbf{x}^k$ and take the k^{th} position in E (line 10).

Reproduction of Upper-Level Optimisation

The reproduction of the upper-level optimisation starts with the generation of upper-level population U. As described in Algorithm 9, n_s upper-level variable vectors $\mathbf{x}_u$ are produced by conducting DE [92] and mutation operators on the upper-level variables X_u of elite population E and archive S_A (lines 3-7). For each $\mathbf{x}_u^i$, N_l lower-level variable vectors $\mathbf{x}_l^{ij}$ are generated by conducting the same operations on the lower-level variables X_l of E and S_A (lines 9-13). These $\mathbf{x}_l^{ij}$ are combined with $\mathbf{x}_u^i$ to form an initial lower-level population L_i (lines 14-15). The upper-level population U consists of n_s different lower-level populations (line 17). The upper-level optimisation runs an independent lower-level optimisation for each L_i, resulting in n_s sets of potentially feasible solutions $\{S_{L1}, \ldots, S_{Ln_s}\}$ (line 18). All solutions in these solution sets are predicted to be feasible on lower-level surrogate h_l. When the optimisation returns from the lower-level to the upper-level optimisation, all potentially feasible solutions are gathered in a set S_U (line 20) and then evaluated on upper-level surrogate h_u. Based on the predicted objective values $h_u(S_U)$, the optimal solution $\mathbf{x}^* \in S_U$ is selected as the output of the upper-level optimisation (line 21). Note that the changes of the upper-level variables (lines 4-7) will not make the final optimisation result in S_U (line 20) violate the constraints, because these changes are made before the lower-level optimisation (line 18).

Algorithm 9 Reproduction

Input:

E: Elite population;
S_A: An archive of evaluated solutions;
N_u: Size of the upper-level population U;
N_l: Size of the lower-level population L;
X_u: Set of upper-level variables;
X_l: Set of lower-level variables;
h_u: Upper-level surrogate;
h_l: Lower-level surrogate.

Procedure:

$n_s = \frac{N_u}{N_l}$
$U = \emptyset$
for $i = 1$ to n_s **do**
  $\mathbf{x}^e \leftarrow$ A solution randomly selected from E.
  $\mathbf{x}^{a1}, \mathbf{x}^{a2} \leftarrow$ 2 solutions randomly selected from S_A.
  $\mathbf{x}_u^i \leftarrow$ DE_Operator$(\mathbf{x}^e, \mathbf{x}^{a1}, \mathbf{x}^{a2}, X_u)$.
  $\mathbf{x}_u^i \leftarrow$ Mutation_Operator$(\mathbf{x}_u^i)$.
  $L_i = \emptyset$
  for $j = 1$ to N_l **do**
    $\mathbf{x}^e \leftarrow$ A solution randomly selected from E.
    $\mathbf{x}^{a1}, \mathbf{x}^{a2} \leftarrow$ 2 solutions randomly selected from S_A.
    $\mathbf{x}_l^{ij} \leftarrow$ DE_Operator$(\mathbf{x}^e, \mathbf{x}^{a1}, \mathbf{x}^{a2}, X_l)$.
    $\mathbf{x}_l^{ij} \leftarrow$ Mutation_Operator$(\mathbf{x}_l^{ij})$.
    $\mathbf{x}^{ij} \leftarrow$ Combine$(\mathbf{x}_u^i, \mathbf{x}_l^{ij})$.
    $L_i = L_i \cup \{\mathbf{x}^{ij}\}$
  end for
  $U = U \cup L_i$
  $S_{Li} \leftarrow$ Lower-Level_Opt(L_i, X_l, h_l, t_{max})./*Potentially feasible solutions in L_i.*/
end for
S_U=$\{L_1, \ldots, L_{n_s}\}$/*Potentially feasible solutions in U.*/
$\mathbf{x}^* \leftarrow$ The optimal solution in $h_u(S_U)$.

Output:

$\mathbf{x}^*$: Optimal solution to be evaluated on expensive functions.

Algorithm 10 Lower-Level_Opt

Input:

L_i^0: Initialised lower-level population;
X_l: Set of lower-level variables;
h_l: Lower-level surrogate;
t_{max}: Maximum iterations of lower-level optimisation.

Procedure:

1: **for** $t = 1$ to t_{max} **do**
2: $L_i^t = L_i^{t-1}$ /*Population in the new generation.*/
3: $h_l(\mathbf{x}) \leftarrow$ Evaluate $\mathbf{x} \in L_i^t$ on h_l.
4: **for** $\mathbf{x} \in L_i^t$ and $h_l(\mathbf{x})$ is infeasible **do**
5: $\mathbf{x}^{r1}, \mathbf{x}^{r2}, \mathbf{x}^{r3} \leftarrow$ Randomly select 3 different solutions from L_i^{t-1}.
6: $\mathbf{x} \leftarrow$ DE_Operator$(\mathbf{x}^{r1}, \mathbf{x}^{r2}, \mathbf{x}^{r3}, X_l)$.
7: $\mathbf{x} \leftarrow$ Mutation_Operator$(\mathbf{x})$.
8: **end for**
9: **end for**
10: $S_{Li} \leftarrow$ Solutions $\mathbf{x} \in L_i^t$ and $h_l(\mathbf{x})$ are feasible.

Output:

S_{Li}: A set of potentially feasible solutions.

Lower-Level Optimisation

For each subpopulation L_i of the upper-level population U, an independent run of the lower-level optimisation is conducted. The pseudo code of the lower-level optimisation is given in Algorithm 10. In the lower-level optimisation, new solutions are generated by conducting DE and mutation operations on the lower-level variables X_l of three parent solutions. These parent solutions are randomly selected from the current generation. The generated new solutions will replace the potentially infeasible solutions in the current generation (lines 5-7). As the generation number t increases, the number of potentially feasible solutions in L_i^t will keep increasing. The search for feasible solutions will be terminated once the maximum number of generation t_{max} is reached. Then the lower-level optimisation outputs all potentially feasible solutions at the final generation (line 10).

4.3 Experimental Studies

In this section, we demonstrate the performance of the proposed SAB-DE framework by conducting several computational experiments.

4.3.1 Performance of Sensitivity Analysis on Solution Feasibility

We first investigate the performance of our sensitivity analysis method, which quantifies the impacts of decision variables on solution feasibility and assists the identification of lower-level variables X_l. Considering the numerical values of real impacts or even the correct division of engine variables is unknown in the real-world engine calibration problem, our experiment is conducted on some benchmark test problems selected from IEEE CEC 2006 special session on constrained real-parameter optimisation [52]. Existing constrained optimisation test problems (from CEC) do not include any suitable benchmark test problems (the feasible region of these benchmark test problems depends on all decision variables). For our first experiment, we need to construct some artificial test problems where their solution feasibility is dependent on a subset of decision variables. In practice, we construct some test problems by selecting three benchmark problems (g01, g07, and g09) and reconstructing their feasible regions with selected constraints. Detailed information about the selected problems and the test problems we construct are summarised as follows (see also Table 4.1):

- g01 is a 13-dimensional problem with 9 linear inequality constraints. We use 6 linear constraints g_1, g_2, g_3, g_4, g_5, and g_6 to construct the feasible region, making their solution feasibility depends on the subset $\{x_1, x_2, x_3, x_{10}, x_{11}, x_{12}\}$. The sets of variables that have impacts and have no impact on solution feasibility are denoted by X_+ and X_-, respectively. For g01, X_+=$\{x_1, x_2, x_3, x_{10}, x_{11}, x_{12}\}$, X_-=$\{x_4, x_5, x_6, x_7, x_8, x_9, x_{13}\}$.

- g09 is a 7-dimensional problem with 4 nonlinear inequality constraints. We use 2 nonlinear constraints g_1, g_2 to construct the feasible region, making their solution feasibility

Table 4.1: Details of the CEC2006 benchmark problems used in the experiments. $|X_+|$, $|X_-|$ are the numbers of decision variables that have impacts or have no impact on solution feasibility, respectively. LI and NI are the numbers of linear or nonlinear inequality constraints, respectively. The numbers in the brackets indicate the number of LI or NI we used to construct the feasible region.

Problems	Variables	$\|X_+\|$	$\|X_-\|$	LI	NI
g01	13	6	7	9(6)	0
g09	7	5	2	0	4(2)
g07	10	6	4	3(1)	5(3)

dependent on the subset $X_+=\{x_1, x_2, x_3, x_4, x_5\}$. Therefore, $X_-=\{x_6, x_7\}$.

- g07 is a 10-dimensional problem with both linear and nonlinear inequality constraints. We use 1 linear constraint g_3 and 3 nonlinear constraints g_4, g_5, g_8 to construct the feasible region. For g07, $X_+=\{x_1, x_2, x_3, x_4, x_9, x_{10}\}$, $X_-=\{x_5, x_6, x_7, x_8\}$.

Despite the use of benchmark problems, it should be noted that, for these problems, the exact impacts of decision variables on solution feasibility are still unknown. For this reason, it is difficult to do a quantitative comparison between our quantified impacts and the numerical values of real impacts. However, as the feasible regions of these problems are dependent on a subset of decision variables X_+, it is clear that the variables in X_+ have some impacts on solution feasibility. Therefore, a qualitative comparison is conducted (between X_+ and X_l) to evaluate the performance of our sensitivity analysis method. Clearly, if our quantified impacts show that a variable has impact on solution feasibility (that is, the probability of being allocated to lower-level variables $> 50\%$), then it will be added to a set X_l. The performance of our method is measured by two indicators about X_l and X_+:

$$\text{Accuracy} = \frac{|X_l \bigcap X_+|}{|X_+|}, \tag{4.12}$$

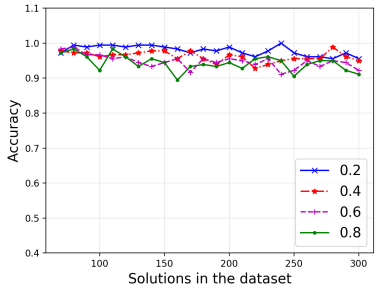
$$\text{Efficiency} = \frac{|X_l \bigcap X_+|}{|X_l \bigcup X_+|}. \tag{4.13}$$

The accuracy indicator counts how many variables in X_+ are identified successfully by our

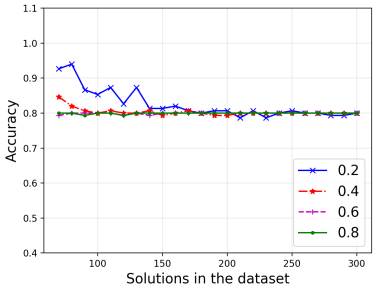
method. The efficiency indicator takes these variables in X_l that have no impact on solution feasibility into account, since allocating these irrelevant variables to X_l will lower the efficiency of handling constraints in the lower-level optimisation.

In SAB-DE, the variable division is based on the result of sensitivity analysis for the evaluated solutions in S_A. In this experiment, a pre-sampled dataset works as a substitute for S_A, where 70 to 300 solutions are randomly sampled from the decision space. As the ratio between the feasible region and the decision space can be different in the optimisation problems to be solved, the performance is investigated under four different situations, where 20%, 40%, 60%, and 80% of the solutions in the sampled dataset are feasible.

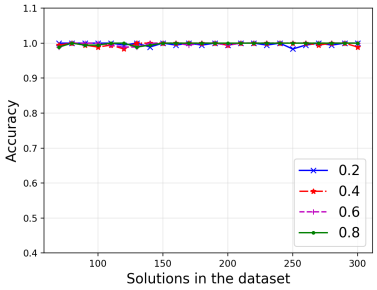
The mean accuracies and efficiencies of sensitivity analysis on modified problems g01, g09, and g07 are plotted in Fig. 4.4. Each result is obtained from 30 independent runs. It can be seen from Figs. 4.4a, 4.4b, and 4.4c that the majority of variables in X_+ have been allocated to X_l successfully on three problems no matter the ratio of feasible solutions or the size of datasets (x-axis). On problem g09, accuracies are close to 0.8 as the variable x_5 in X_+ has a very weak impact on solution feasibility. Therefore, our method tends to treat x_5 as a variable that is irrelevant to the solution feasibility of g09. By comparison, in Figs 4.4d, 4.4e, and 4.4f, for a fixed ratio of feasible solutions, the efficiency increases (until reaches its maximum) as the size of the given dataset increases, which is caused by the simultaneous increase in the number of feasible solutions. Also, the efficiency comparison between four feasible ratios indicates that a large amount of feasible solutions is beneficial to the efficient division of variables. When only a few feasible solutions are available, the variables in X_- may be allocated to X_l, which will lower the efficiency of constraint handling. This also explains why the accuracy of ratio 0.2 in Fig. 4.4b is higher than other ratios at the early stage. In summary, the results of qualitative comparison on three different problems have demonstrated that the proposed sensitivity analysis method is effective. The quantified impacts lead to accurate division of variables, where most variables in X_+ are identified as lower-level variables X_l.



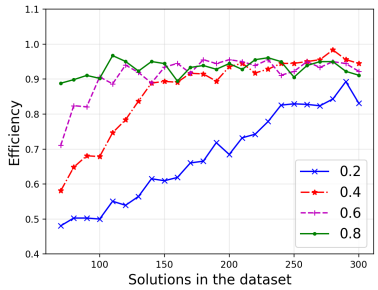
(a) g01: Linear constraints



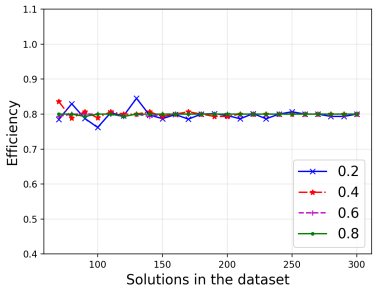
(b) g09: Nonlinear constraints



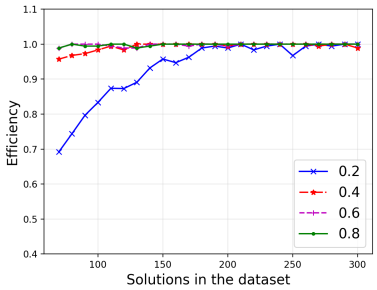
(c) g07: Linear and nonlinear constraints



(d) g01: Linear constraints



(e) g09: Nonlinear constraints



(f) g07: Linear and nonlinear constraints

Figure 4.4: The accuracy (upper row) and efficiency (lower row) of sensitivity analysis on modified problems g01, g09, and g07. Each result is the mean value over 30 independent runs. The size of the randomly sampled dataset used for sensitivity analysis varies from 70 to 300 with 10 intervals. The ratio between feasible solutions and all solutions in the dataset is set to 0.2, 0.4, 0.6, and 0.8.

4.3.2 Performance of Ordinal Regression Surrogate

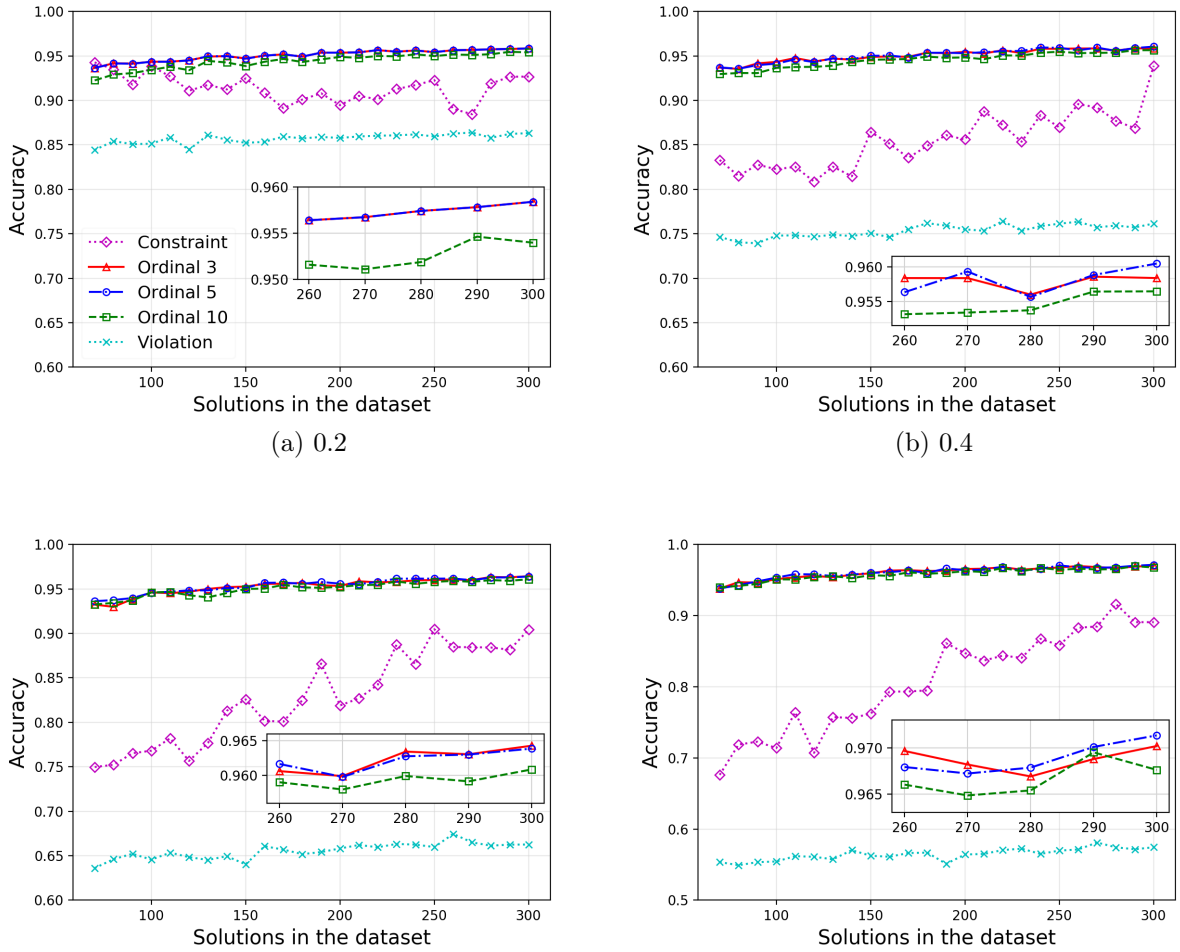
In the lower-level optimisation, we use an ordinal-regression-based surrogate to estimate the ordinal landscape of constraints, which simplifies the architecture of lower-level optimisation without damaging the prediction accuracy of solution feasibility. In this section, we provide empirical evidence to show the effectiveness and efficiency of our ordinal-regression-based surrogate.

The empirical experiment is conducted on an experience-based gasoline engine model, which is learned from real engine data [123]. 70 to 300 random solutions are sampled from the engine model as the training dataset, and an extra 10000 random solutions are sampled

as the test dataset. To obtain a comprehensive understanding of surrogate modelling performance, four different situations are considered, where approximately 20%, 40%, 60%, and 80% of the solutions in the training and test datasets are feasible. Two constraint modelling methods are compared with our ordinal-regression-based Kriging surrogates for 30 independent runs. The first method uses four Kriging surrogates to approximate the value of four constraints separately. If the predictions of all four surrogates satisfy their corresponding constraints, then a solution will be predicted to be feasible. The second method uses one Kriging surrogate to estimate the sum of violations in all constraints. A solution is predicted to be feasible if the estimated sum of its constraint violations is less than or equivalent to 0. For the sake of convenience, the two comparison methods mentioned above are denoted as 'Constraint' and 'Violation' in Fig. 4.5. Additionally, we train ordinal-regression-based surrogates with three different ordinal levels n_o=$\{3, 5, 10\}$. All Kriging surrogates are implemented using DACE [83], the parameter range of the correlation function is $\theta \in [10^{-5}, 100]$, $p = 2$. The surrogate modelling performance is evaluated through the prediction accuracy of solution feasibility and the runtime of using surrogates.

Fig. 4.5 shows the prediction accuracy of the solution feasibility on the engine calibration problem. For ordinal-regression-based surrogates, a stable enhancement of prediction accuracy can be observed as the size of the training dataset increases. The comparison between three ordinal-regression-based surrogates shows that the difference between their prediction accuracies is less than 1% (see the inset diagrams in Fig. 4.5). Hence, the prediction accuracy of ordinal-regression-based surrogates is not particularly sensitive to the number of ordinal levels n_o.

For the compared constraint modelling methods, it can be seen that their overall prediction accuracies are decreasing as the ratio of feasible solutions increases. This is explainable since both the combination of constraint regressors and the regressor of total constraint violations are inclined to produce infeasible predictions. When the ratio of feasible solutions is greater than 0.2, the prediction accuracy of the combination of four constraint regressors



(a) 0.2 (b) 0.4

(c) 0.6 (d) 0.8

Figure 4.5: The prediction accuracy of solution feasibility on the engine calibration problem. Ordinal 3, 5, 10 indicates the ordinal-regression-based surrogates trained with n_o=$\{3, 5, 10\}$. Each result is the mean of results collected from 30 independent runs. The size of the randomly sampled training dataset varies from 70 to 300 with 10 intervals. A randomly sampled test dataset of size 10000 is used to measure the prediction accuracy. The ratios between feasible solutions and all solutions in both the training and test datasets are set to approximately 0.2, 0.4, 0.6, and 0.8.

is increasing as the size of the training dataset increases. In contrast, the prediction accuracy of the constraint violation regressor enhances at a slow rate. The comparison between three kinds of surrogates has demonstrated that our ordinal-regression-based surrogate has higher accuracy than the compared two constraint modelling surrogates. These results indicate that our ordinal-regression-based surrogate is effective in the prediction of solution feasibility.

A runtime comparison is also conducted to compare the efficiency of three kinds of surrogates. Fig. 4.6 presents the time cost of surrogate training and prediction over different training dataset sizes. It can be observed that the time cost of using the combination of four

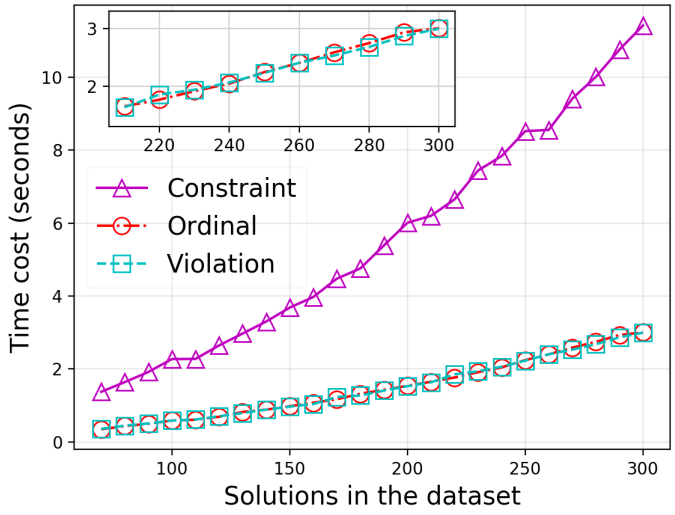


Figure 4.6: The time cost of three kinds of surrogates over different training dataset sizes. Each result is the mean value over 30 independent runs. The size of the randomly sampled training dataset varies from 70 to 300 with 10 intervals. The runtime includes the time cost of surrogate training and prediction.

constraint regressors is three times more than the cost of maintaining only one surrogate (an ordinal-regression-based surrogate or a surrogate of constraint violations). The efficiency of our ordinal-regression-based surrogate has been demonstrated.

4.3.3 Comparison of Environmental Selection Strategies

An ϵ-selection strategy is developed to select evaluated solutions for maintaining elite population E. In order to investigate the effect of the ϵ-selection strategy in SAB-DE and the influence of different ϵ parameter setups, we compare three SAB-DE variants in this section:

- SAB-DE with $\frac{1}{3}$-selection strategy. In this variant, top-$\frac{1}{3}$ infeasible solutions have a chance to be selected for the elite population E.

- SAB-DE with $\frac{2}{3}$-selection strategy. In this variant, top-$\frac{2}{3}$ infeasible solutions are considered for the elite population E.

- SAB-DE with 0-selection strategy. In this variant, only feasible solutions can be selected for the update of the elite population E. Thus infeasible solutions cannot assist the evolutionary search.

The comparison is conducted on the engine model mentioned in Section 4.3.2. As suggested in [46], the size of initial dataset N_{init} is $11d-1$, which is 65 in this engine calibration problem. The total evaluation budget FE_{max} is 300. For the upper-level optimisation, the size of the elite population N_e and that of upper-level population N_u are set to 20 and 4000, respectively. For the lower-level optimisation, the size of lower-level population N_l and the maximum generation t_{max} of the lower-level optimisation are 40 and 20, respectively. n_o is set to 10, the parameters of Kriging surrogates $\theta \in [10^{-5}, 100]$, p =2. The parameters of DE operator and polynomial mutation operator are $F = 0.5, CR = 1.0$ and $p_m = \frac{1}{d}, \eta = 20$, respectively, as suggested in [120].

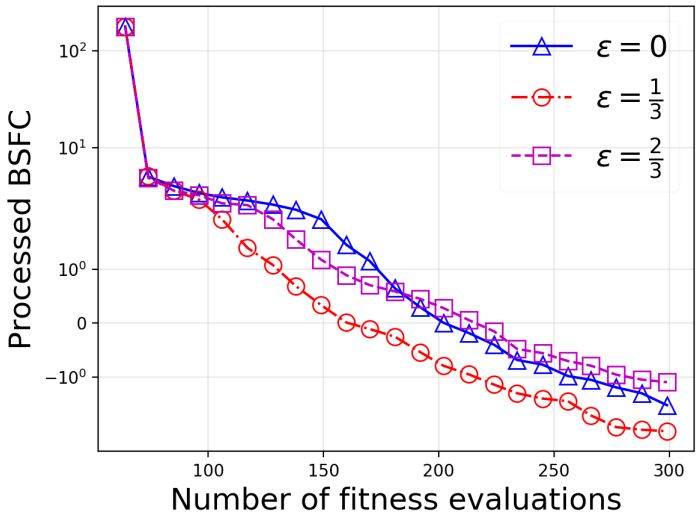


Figure 4.7: Mean BSFC values obtained by SAB-DE with different ϵ setups over 20 independent runs. The smallest BSFC value is reached when SAB-DE employs the $\frac{1}{3}$-selection strategy.

Mean BSFC values obtained by three SAB-DE variants over 20 independent runs are plotted in Fig. 4.7. Since the raw engine data is confidential and we are not allowed to disclose it, all engine calibration results (BSFC) have been processed to hide their original values. It can be observed that using $\frac{1}{3}$-selection strategy in SAB-DE leads to the smallest BSFC optimisation result in this engine calibration problem. Compared with other ϵ setups, setting ϵ to $\frac{1}{3}$ leads to a quicker decrease of BSFC value when 100 to 150 evaluations are used. The comparison between the results of $\epsilon = 0$ and $\epsilon = \frac{1}{3}$ has demonstrated the effectiveness of exploiting the infeasible solutions that are very close to the feasible region. However, the comparison between the results of $\epsilon = \frac{1}{3}$ and $\epsilon = \frac{2}{3}$ also shows that a small ϵ value is preferable to a larger ϵ value. When ϵ is large, some infeasible solutions that are not close to the feasible region will be considered for elite population E, which is not desirable. Consequently, we set $\epsilon = \frac{1}{3}$ as a default setup for SAB-DE.

4.3.4 Parameter Tuning

To investigate the effect of the size of elite population E on the performance of SAB-DE, we run SAB-DE on the engine calibration problem with three different setups ($|E| = 10, 20, 30$) for 20 independent runs. It turns out that SAB-DE achieves competitive BSFC values when $|E|$ is set to 10 or 20. The result obtained when E=30 is larger than the results obtained by other setups. Considering the DE operation in the upper-level optimisation uses the solutions randomly selected from the archive S_A (see Algorithm 9), the diversity of solutions generated in the upper-level optimisation can be guaranteed. Therefore, it is not necessary to use a very large elite population to ensure the diversity of reproduction. In contrast, the convergence speed of the upper-level optimisation will be slow if a very large elite population is used in SAB-DE. Hence, in Section 4.3.5, the size of elite population $|E|$ is set to 20.

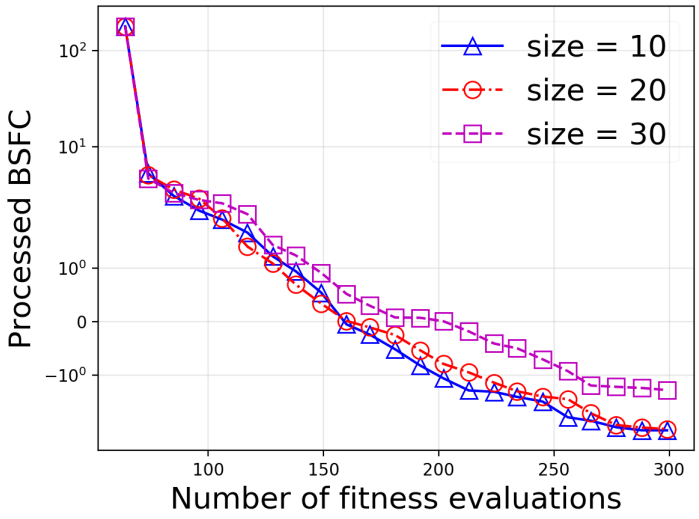


Figure 4.8: Mean BSFC values obtained by SAB-DE with different elite population sizes over 20 independent runs. SAB-DE achieves competitive BSFC results when the elite population size is set to 10 or 20.

4.3.5 Performance on Engine Calibration Problems

In this subsection, we compare SAB-DE with four state-of-the-art algorithms:

- Two constrained optimisation algorithms used in the engine industry [123]: A Bayesian optimisation method derived from EGO but designed to handle constraints (denoted by cons_EGO) and a self-adaptive GA customised for this calibration problem (denoted by adaptiveGA). cons_EGO and adaptiveGA have been fine-tuned on the same engine calibration problem in [123].
- MPMLS [50]: A SAEA which uses multiple penalties and multiple local surrogates to handle constraints.
- PC-EGO [73]: A parallel constrained EGO.

The settings of the comparison algorithms are the same as suggested in [123, 50, 73] except that the population size of MPMLS is set to 20, since the original population size in MPMLS leads to poor optimisation results on our engine calibration problem. No additional modifi-

cations have been made to the parameter setups of PC-EGO, because the suggested setups are optimal on our engine calibration problem. Other experimental setups are the same as described in Section 4.3.3.

Calibration Results

Mean BSFC values obtained by SAB-DE and the four comparison algorithms over 30 independent runs are depicted in Fig. 4.9. Again, BSFC optimisation results have been processed (based on the result of cons_EGO) to hide their original values. It can be observed that

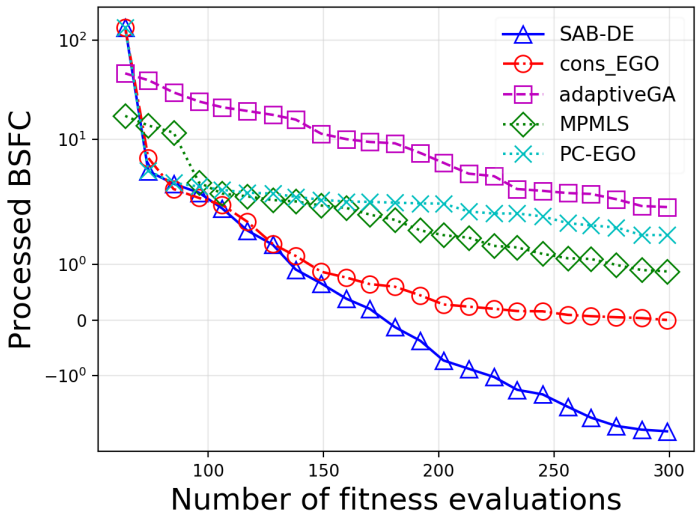


Figure 4.9: Mean BSFC values obtained by SAB-DE and four comparison algorithms over 30 independent runs. Since the BSFC values obtained by all algorithms decrease drastically at an early stage, the y-axis is displayed on an exponential scale. It can be seen that SAB-DE outperforms four comparison algorithms in terms of the optimal BSFC reached. SAB-DE saves approximately 120 expensive evaluations than comparison algorithms.

SAB-DE outperforms four comparison algorithms in terms of the optimal BSFC reached. Clearly, on average, the optimal BSFC value obtained by SAB-DE is smaller than that obtained by adaptiveGA, MPMLS, and PC-EGO when more than 70 solutions have been evaluated. In comparison, SAB-DE and cons_EGO have reached comparable BSFC values when the number of evaluations is less than 130. However, after that, the difference between

Table 4.2: Statistical results of optimal BSFC values obtained by SAB-DE and four comparison algorithms. 30 independent runs are conducted to obtain mean results. '+', '≈', and '−' denote SAB-DE is statistically significantly superior to, almost equivalent to, and inferior to the compared algorithms in the Wilcoxon rank sum test (significance level is 0.05), respectively. The best result is highlighted.

Algorithms	BSFC	+/ ≈/ -
SAB-DE	-2.0293E+00± 2.0254E+00	
cons_EGO	0.0000E+00 ± 1.0371E+00	+
adaptiveGA	2.1038E+00 ± 1.4969E+00	+
MPMLS	8.7745E-01 ± 1.3827E+00	+
PC-EGO	1.5339E+00 ± 9.9921E-01	+

SAB-DE and cons_EGO appears and such a difference is increasing until the evaluation budget has run out.

Considering this engine calibration problem is a real-world application, it is necessary to evaluate whether the improvement in BSFC values caused by using SAB-DE is statistically significant and is meaningful to industry. The result of a statistical test is reported in Table 4.2. It shows that the use of SAB-DE results in a significant improvement in BSFC values from the perspective of statistics. However, from the view of industry, since all BSFC values have been processed to hide their original values, it is hard to explain the meaning of such an improvement using BSFC values. Alternatively, we can explain the advantage of SAB-DE using the cost of expensive evaluations. As illustrated in Fig. 4.9, for SAB-DE, it costs about 180 evaluations to reach the smallest BSFC value achieved by comparison algorithms. Therefore, in contrast to using comparison algorithms, the use of SAB-DE saves at least 120 expensive fitness evaluations on engine facilities. It should be noted that, in engine calibration problems, a real engine performance evaluation on engine facilities is very costly in terms of both time and financial budget [116]. Saving 120 evaluations indicates a shorter engine development cycle than before and a saving of millions of dollars [59]. Therefore, the improvement caused by using SAB-DE is meaningful to industry.

Efficiency Analysis

Further comparisons are conducted to evaluate the efficiency of SAB-DE. Fig. 4.10 displays the number of feasible solutions obtained by SAB-DE and comparison algorithms. It can be

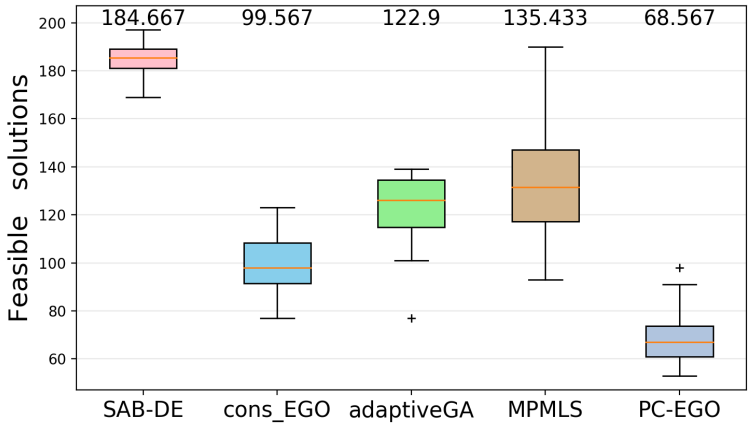


Figure 4.10: Statistical results of the number of feasible solutions obtained by SAB-DE and four comparison algorithms over 30 independent runs. Mean values are shown on the top. SAB-DE finds much more feasible solutions than comparison algorithms, showing a high efficiency of searching for the feasible region.

seen that SAB-DE finds more feasible solutions than comparison algorithms, implying the bilevel architecture in SAB-DE enhances the efficiency of searching for the feasible region.

The computational efficiency of SAB-DE is also investigated in the comparison between the best two SAEAs in Fig. 4.9. Note that cons_EGO and the gasoline engine model are implemented in MATLAB and SAB-DE is implemented in Python. Therefore, the absolute runtimes of two SAEAs are not directly comparable. Alternatively, we compare the number of operations conducted by SAB-DE and cons_EGO between two expensive evaluations. As illustrated in Fig. 4.11, SAB-DE conducts fewer operations than cons_EGO. Clearly, since the constraint handling process in SAB-DE is pertinent to the lower-level variables, it is not necessary for SAB-DE to generate as many candidate solutions as cons_EGO. In addition, reproductive operations, such as DE and mutation operations, will be conducted on only upper-level variables in the upper-level optimisation. The reproductive operations

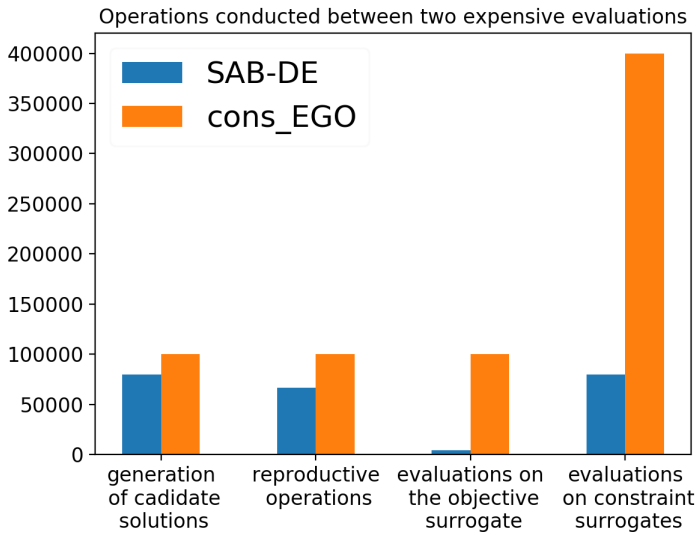


Figure 4.11: Numbers of the operations conducted by SAB-DE and cons_EGO in an iteration (between two expensive evaluations). Note these numbers are compared through their upper bounds, since the actual numbers are dependent on the number of potential feasible solutions found in an iteration.

in SAB-DE are mainly conducted on lower-level variables and the total reproductive operations are less than that in cons_EGO. Likewise, the evaluations on the objective surrogate only happen in the upper-level optimisation, thus SAB-DE conducts a few operations in this stage. Finally, for the evaluations on constraint surrogates, as explained in Section 4.3.2, the use of our ordinal-regression-based surrogate saves plenty of surrogate evaluation operations. Consequently, the computational efficiency of SAB-DE is higher than that of cons_EGO.

4.3.6 Summary

Our computational studies have provided empirical evidence to support the contributions we claimed in Section 1.4. First, three constrained benchmark test problems are selected and then reconstructed to make their feasible regions affected by a subset of decision variables. A qualitative comparison has been conducted on these problems to demonstrate the accuracy and efficiency of dividing variables with quantified impacts. Second, we have demonstrated the effectiveness of using an ordinal-regression-based Kriging model as a constraint surrogate.

The comparison between ordinal-regression-based surrogates and other constraint modelling methods shows that using the our surrogate will not lower the performance of predicting solution feasibility. Furthermore, our surrogate tends to be more computationally efficient when compared with the compared methods. Third, the effect of different ϵ-selection strategies has been investigated. Better optimisation results are achieved when a $\frac{1}{3}$-selection strategy is applied to SAB-DE. It is beneficial to exploit the infeasible solutions that are very close to the feasible region. Fourth, the optimisation performance of SAB-DE has been demonstrated on a real-world engine calibration problem. SAB-DE is compared with four state-of-the-art constrained optimisation algorithms. The calibration results show that SAB-DE outperforms the compared algorithms. A significant improvement in BSFC is achieved, which is meaningful to this real-world application. Besides, compared with other algorithms, SAB-DE is efficient when searching for feasible solutions. Additionally, the advantage of SAB-DE in computational efficiency is also discussed.

4.4 Chapter Summary

In this chapter, we propose a surrogate-assisted bilevel differential evolution, SAB-DE, to solve the expensive constrained optimisation problems in engine calibration applications. The bilevel architecture used (not to be confused with the traditional bilevel optimisation) facilitates constraint handling and optimisation. The lower-level focuses on tuning lower-level variables to handle constraints, while the upper-level adjusts upper-level variables to minimise the objective. A PCA-based sensitivity analysis method is developed to quantify the impact of decision variables on solution feasibility. The variables with significant impact on solution feasibility are allocated to lower-level variables, improving the efficiency of constraint handling in the lower-level. To simplify the architecture of the lower-level optimisation and reduce the time cost of using surrogates, an ordinal-regression-based surrogate is adapted to approximate the ordinal landscape of multiple constraints. Additionally, infea-

sible solutions are allowed to assist the evolutionary search through an ϵ-selection strategy. Our computational studies have demonstrated that the proposed optimisation algorithm is effective and efficient on a real-world engine calibration problem. Compared with state-of-the-art engine calibration methods, better BSFC values are achieved and more feasible solutions are detected.

However, the surrogates used in SAB-DE are derived from Kriging models whose computational cost will increase rapidly as the number of decision variables increases. Hence, high computational cost can be a limitation and will prevent SAB-DE from being applied to large-scale ECOPs. Besides, if all decision variables of an ECOP have significant impacts on solution feasibility, then SAB-DE will be inclined to classify all decision variables into lower-level variables. As a result, the advantage of using bilevel architecture in SAB-DE might disappear, indicating a lower efficiency of handling constraints.

The contributions of this chapter can be summarised as:

- A sensitivity analysis method is developed to quantify the impact of decision variables on solution feasibility. This analysis method employs principal component analysis (PCA) to analyze the distribution of feasible solutions. The quantified impacts will be used to guide the division of upper-level and lower-level variables. The decision variables that have significant impacts on solution feasibility are identified as lower-level variables, while the remaining variables are denoted by upper-level variables. This is an answer to the first question in Section 1.3.2.

- An ordinal-regression-based surrogate is adapted to approximate the ordinal landscape of constraints. Instead of approximating each constraint function with a regression-based surrogate separately, we use only one ordinal-regression-based surrogate to approximate the domination-based ordinal landscape of multiple constraints. The use of our ordinal-regression-based surrogate simplifies the architecture of lower-level optimisation and it is more efficient than using multiple regression-based surrogates to

approximate constraints separately. Moreover, compared with classification-based surrogates, the ordinal-regression-based surrogate also exploits the difference between infeasible solutions. This is an answer to the second question in Section 1.3.2.

- The framework of a novel SAEA (Surrogate-Assisted Bilevel Differential Evolution, SAB-DE) is proposed to solve ECOPs with a bilevel architecture. In the SAB-DE framework, the lower-level aims at optimising lower-level variables to make candidate solutions feasible, while the upper-level focuses on adjusting upper-level variables to optimise the objective. Such a bilevel architecture ensures that more constraint handling effort can be made on lower-level variables. Thus, it is more efficient than a majority of existing constraint handling methods which allocate effort evenly to every decision variable. In comparison, existing studies that use bilevel architectures are designed for bilevel optimisation problems instead of ECOPs. Additionally, a novel environmental selection strategy is employed to exploit the infeasible solutions that are close to the feasible region. This is an answer to the last question in Section 1.3.2.

Chapter 5

Experience-Based Evolutionary Algorithms for Expensive Optimisation

In the last two chapters, we have developed several new SAEAs to solve EMOPs and ECOPs, respectively. We have demonstrated that when solving expensive optimisation problems, our proposed optimisation algorithms are capable of finding near-optimal solutions with a given limited fitness evaluation budget. Significant improvements on optimisation results are observed in our experimental studies. However, when compared with human optimisers, these optimisation algorithms are not intelligent enough. A human being would gain more experience through problem-solving, which helps her/him in solving a new unseen problem. Yet an optimisation algorithm never gains any experience by solving more problems. In this chapter, we continue our research on the topic of expensive optimisation problems. Unlike previous chapters which focus on the target optimisation problems, in this chapter, we take other related optimisation problems into consideration. We aim to make existing optimisation algorithms more intelligent than before through learning experience from related optimisation problems.

In recent years, there have been efforts made towards endowing optimisation algorithms with some abilities of experience learning, which are regarded as experience-based

optimisation. In this chapter, we argue that hard optimisation problems could be tackled efficiently by making better use of the experience gained in related problems. We demonstrate our ideas in the context of expensive optimisation, where we aim to find a near optimal solution to an expensive optimisation problem with as few fitness evaluations as possible. In practice, we propose an experience-based SAEA framework to enhance the optimisation efficiency when applied to expensive problems, where useful experience is learned across related expensive tasks via a novel meta-learning method. The learned experience serves as the task-independent parameters of a deep kernel learning surrogate, then the solutions sampled from the target task are used to further adapt task-specific parameters for the surrogate. With the help of experience learning, competitive regression-based surrogates can be initialised with a cost of only $1d$ solutions from the target task. In Section 5.4, our experimental results demonstrate that the experience learned from related tasks is beneficial to the saving of evaluation budgets on the target problem, indicating that our SAEA framework is suitable for the optimisation of problems whose evaluation is extremely expensive.

5.1 Motivation

When solving a new unseen problem, a human being often benefits from the experience gained from the related problems he/she has seen in the past. Such an ability of experience learning enhances the efficiency of solving new problems and thus is desirable for intelligent optimisation algorithms. To endow optimisation algorithms with some abilities of experience learning, many efforts have been made to combine diverse experience learning techniques with optimisation algorithms, which results in experience-based optimisation algorithms [57, 96, 95]. In the past decade, experience-based optimisation approaches such as evolutionary transfer optimisation [95, 41, 40] and evolutionary multi-tasking optimisation (EMTO) [108, 2, 113] have been proposed to solve diverse optimisation problems, including automatic parameter tuning problems [96], dynamic optimisation problems [78, 79]. These studies

have demonstrated that the idea of experience-based optimisation is effective when solving hard optimisation problems, the experience gained from past problems could be helpful for solving new unseen optimisation problems.

The motivation of this chapter is to demonstrate that the idea of experience-based optimisation is also working in the context of expensive optimisation problems. In other wards, we want to demonstrate that useful experience could be learned from related expensive optimisation problems via using suitable experience learning techniques, and new unseen expensive optimisation problems could be tackled efficiently by using the learned experience in existing SAEAs. In real-world applications, many expensive optimisation problems are related since they are working on similar issues in the same domain. For example, in the domain of gasoline engine calibration, many calibration problems can be treated as related tasks. Although the categories of gasoline engines to be calibrated are different, the physical properties of gasoline will not change. Moreover, the mechanical structures of some gasoline engines are similar [116]. These domain-specific features make many calibration problems related to each other. It is desirable to explore the domain-specific features from related tasks and then use them as experience in SAEAs. The learned experience could help SAEAs to build reliable surrogates with a fewer cost of fitness evaluations than before (e.g., $1d$ evaluations). Consequently, these experience-based SAEAs can save more fitness evaluations (in surrogate initialisation) than non-experience-based SAEAs and thus are more suitable for very expensive optimisation problems, where only 150 or fewer fitness evaluations are allowed during the optimisation.

The related tasks considered in this chapter are expensive and each of them can provide only a small dataset of evaluated samples for experience learning. Therefore, our experience-based SAEA is based on the context of few-shot problems [5, 105], where plenty of small related tasks are available for experience learning. A challenge is that most existing experience-based optimisation approaches cannot learn experience from small related tasks. However, recently, meta-learning [35] has been proved to be powerful in solving few-shot

problems. In meta-learning, the underlying common features of related tasks are extracted as domain experience, which can be integrated with the solutions sampled from new tasks and thus enhance the learning efficiency for new tasks. Quite a few meta-learning methods [105] have been proposed for few-shot problems. Benefiting from the ability of experience learning, these methods are capable of fitting accurate classification or regression models with limited samples from the target task, which motivates us to develop meta-learning methods to learn experience for SAEAs. In this chapter, we propose an experience-based SAEA framework to solve expensive optimisation problems, including multi-objective optimisation and constrained optimisation.

5.2 Preliminaries

A meta deep kernel learning modelling method is developed to learn experience in our SAEA framework. This section gives preliminaries about meta-learning and deep kernel learning. The former is the method of experience learning, the latter is the underlying structure of experience representation.

Meta-Learning in Few-Shot Problems

In the context of few-shot problems, we have plenty of related tasks, each task T contributes a couple of small datasets $D = \{(S, Q)\}$, namely support dataset S and query dataset Q, respectively. After learning from datasets of random related tasks, a support set S_* from new unseen task T_* is given and one is asked to estimate the labels or values of a query set Q_*. The problem is called 1-shot or 5-shot when only 1 data point or 5 data points are provided in S_*. A comprehensive definition of few-shot problems is available in [5, 105].

Meta-learning methods have been widely used to solve few-shot problems [105]. They learn domain-specific features that are shared among related tasks as experience, such experience is used to understand and interpret the data collected from new tasks encountered in

the future.

Deep Kernel Learning

Deep kernel learning (DKL) [110] aims at constructing kernels that encapsulate the expressive power of deep architectures for Gaussian processes (GPs) [109]. To create expressive and scalable closed form covariance kernels, DKL combines the non-parametric flexibility of kernel methods and the structural properties of deep neural networks. A brief introduction to the GP (Kriging) model and the feedforward neural network can be found in Chapter 2. In practice, a deep kernel $k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma})$ transforms the inputs $\mathbf{x}$ of a base kernel $k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\theta})$ through a non-linear mapping given by a deep architecture $\phi(\mathbf{x}|\mathbf{w}, \mathbf{b})$:

$$k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma}) = k(\phi(\mathbf{x}^i|\mathbf{w}, \mathbf{b}), \phi(\mathbf{x}^j|\mathbf{w}, \mathbf{b})|\boldsymbol{\theta}), \tag{5.1}$$

where $\boldsymbol{\theta}$ and $(\mathbf{w}, \mathbf{b})$ are parameter vectors of the base kernel and the deep architecture, respectively. $\boldsymbol{\gamma} = \{\boldsymbol{\theta}, \mathbf{w}, \mathbf{b}\}$ is the set of all parameters in this deep kernel. Note that in DKL, all parameters $\boldsymbol{\gamma}$ of a deep kernel $k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma})$ are learned jointly by using the log marginal likelihood function of GPs as a loss function. Such a jointly learning strategy has been shown to make a DKL algorithm outperform a combination of a deep neural network and a GP model, where a trained GP model is applied to the output layer of a trained deep neural network [110].

Advantages of Applying Meta-Learning to DKL

An important distinction between DKL algorithms and applications of meta-learning to the DKL is that DKL algorithms learn their deep kernels from single tasks instead of collections of related tasks. Such a difference alleviates two drawbacks of single task DKL [101]:

- First, the scalability of deep kernels is no longer an issue as each dataset in meta-learning is small.

- Second, the risk of overfitting is decreased since diverse data points are sampled across tasks.

5.3 Experience-Based Surrogate-Assisted Evolutionary Algorithm Framework

In this section, expensive optimisation is carried out in the context of few-shot problems [96], where the expensive optimisation problem to be solved is denoted as target task T_*, and plenty of small datasets D_i sampled from related tasks T_i are available for experience learning. We propose a SAEA framework to learn experience from T_i and use experience in T_*, which saves fitness evaluations for expensive optimisation problems. A meta deep kernel learning (MDKL) modelling method is developed to learn experience from T_i and then initialises surrogates with a cost of $1d$ evaluations on T_*. Our SAEA framework combines MDKL with existing regression-based SAEAs. A new update strategy is designed to maintain and adapt surrogates for SAEAs. As a result, although many fewer evaluations from T_* are used, our SAEA framework can still achieve competitive or even better optimisation results than the SAEAs that are unable to learn experience from T_i.

5.3.1 Overall Working Mechanism

A diagram of our experience-based SAEA framework is illustrated in Fig. 5.1. All modules covering the evolutionary optimisation of target task T_* are included in a grey block. The modules beyond the grey block are associated with related tasks T_i and experience learning, which are the main distinctions between our SAEA framework and existing SAEAs. The MDKL surrogate modelling method used in our SAEA framework consists of two procedures: meta-learning procedure and adaptation procedure. The former learns experience from T_i, and the latter uses experience to adapt surrogates to approximate T_*.

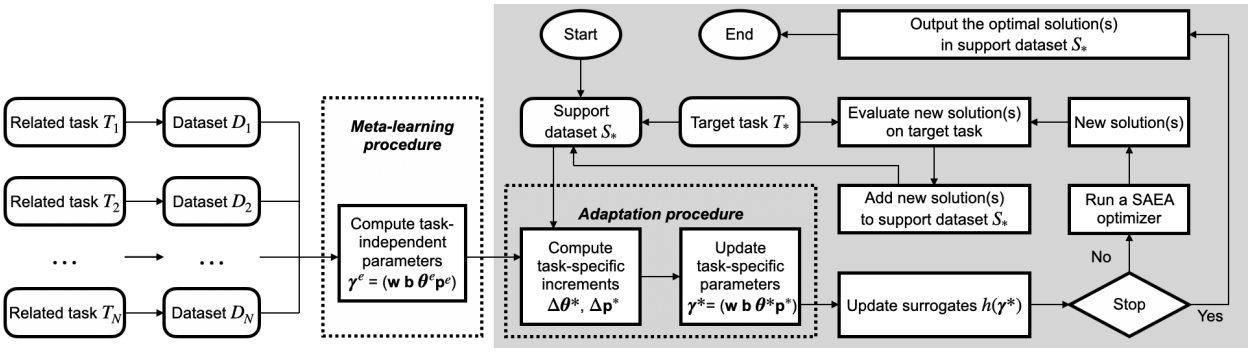


Figure 5.1: Diagram of the proposed experience-based SAEA framework. The grey block includes all modules that are related to the evolutionary optimisation of target task T_*. The MDKL surrogate modelling method used in the framework consists of a meta-learning procedure and an adaptation procedure. The meta-learning procedure learns experience (task-independent parameters $\boldsymbol{\gamma}^e$) from related tasks T_i. Based on the learned experience, the adaptation procedure adapts MDKL task-specific parameters to approximate target task T_*. Note that existing SAEAs train and update their surrogates on S_* only, thus their workflows do not contain a meta-learning procedure and they cannot gain experience from related tasks.

The framework of experience-based evolutionary optimisation is depicted in Algorithm 11, it consists of the following major steps:

1. **Experience learning**: Before the evolutionary optimisation starts, a meta-learning procedure is conducted to train task-independent parameters $\boldsymbol{\gamma}^e$ for MDKL surrogates (line 1). $\boldsymbol{\gamma}^e$ are trained on N_m datasets $\{D_{m1}, \ldots, D_{mN_m}\}$ which are collected from N related tasks $\{T_1, \ldots, T_N\}$. The learned parameters $\boldsymbol{\gamma}^e$ are the experience that represents the domain-specific features of related tasks.

2. **Initialise surrogates with experience**: Evolutionary optimisation starts when a target optimisation task T_* is given. An initial dataset S_* is sampled (line 2) to adapt task-specific parameters $\boldsymbol{\gamma}^*$ on the basis of experience $\boldsymbol{\gamma}^e$. After that, MDKL surrogates are updated (line 3).

3. **Reproduction**: MDKL surrogates $h(\boldsymbol{\gamma}^*)$ are combined with a SAEA optimizer *Opt* to search for optimal solution(s) $\mathbf{x}^*$ on $h(\boldsymbol{\gamma}^*)$ (line 6). This is implemented by replacing the original (regression-based) surrogates in a SAEA with $h(\boldsymbol{\gamma}^*)$.

Algorithm 11 Experience-Based SAEA Framework.

Input:

D_i: Datasets collected from related tasks T_i, i=$\{1, \ldots, N\}$;
N_m: Number of subsets D_m for meta-learning;
$|D_m|$: Size of subsets D_m. $|D_m| \leq |D_i|$ due to $D_m \subseteq D_i$.
B: Number of related tasks in a batch;
α, β: Surrogate learning rates;
T_*: Target task;
Opt: A SAEA optimiser;
FE_{max}: Fitness evaluation budget.

Procedure:

1: Experience $\boldsymbol{\gamma}^e \leftarrow$ Meta-learning$(D_i, N_m, |D_m|, B, \alpha)$. /*Algorithm 12.*/
2: $S_* \leftarrow$ Sampling $1d$ solutions from T_*.
3: $h(\boldsymbol{\gamma}^*) \leftarrow$ Adaptation$(\boldsymbol{\gamma}^e, S_*, \beta)$. /*Initialise surrogate, Algorithm 13.*/
4: Set evaluation counter $FE = |S_*|$.
5: **while** $FE < FE_{max}$ **do**
6: Candidate solution(s) $\mathbf{x}^* \leftarrow$ Surrogate-assisted optimisation $(Opt, h(\boldsymbol{\gamma}^*))$
7: $f(\mathbf{x}^*) \leftarrow$ Evaluate $\mathbf{x}^*$ on T_*.
8: $S_* \leftarrow S_* \cup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$.
9: $h(\boldsymbol{\gamma}^*) \leftarrow$ Update$(\boldsymbol{\gamma}^*, S_*, \beta)$. /*Algorithm 14.*/
10: Update FE.
11: **end while**

Output: Optimal solutions in S_*.

4. **Update archive**: New optimal solution(s) $\mathbf{x}^*$ is evaluated on target task T_* (line 7) and the evaluated solutions will be added to dataset S_* (line 8). S_* serves as an archive.

5. **Update surrogates**: Since new data has been added to S_*, further surrogate adaptation is triggered. As a result, surrogates $h(\boldsymbol{\gamma}^*)$ are updated (line 9).

6. **Stop criterion**: Once the evaluation budget has run out, the evolutionary optimisation will be terminated and the optimal solutions in dataset S_* will be outputted. Otherwise, the algorithm goes back to step 3.

In the following subsections, we first present the details of our MDKL surrogate modelling method, including how to learn experience through through meta-learning and adapt it to a specific target task. Then we explain the surrogate update strategy in our experience-based SAEA framework. Finally, we discuss the usage of MDKL surrogates and the compatibility

of our SAEA framework with existing SAEAs.

5.3.2 Learning and Using Experience by Meta Deep Kernel Learning

In MDKL, the domain-specific features of related tasks are used as experience, which are represented by the task-independent parameters $\boldsymbol{\gamma}^e$ learned across related tasks. To make MDKL more capable of expressing complex domain-specific features, the base kernel $k(\mathbf{x}^i, \mathbf{x}^j | \boldsymbol{\theta})$ in GP is combined with a neural network $\phi(\mathbf{w}, \mathbf{b})$ to construct a deep kernel (see Eq.(5.1)).

The main novelties of our MDKL surrogate modelling method can be summarised as follows:

- A simple and efficient meta-learning structure for experience learning. Our method learns only one common neural network for all related tasks, such a neural network works as a component of the shared deep kernel. In contrast, [25, 101] generate separate deep kernels for each related task and train multiple neural networks to encode and decode task features, which makes their model complexity grow with the number of related tasks.

- The explicit task-specific adaptation for the deep kernel, which is implemented by cumulating task-specific increments on the basis of the learned experience.

The modelling of a MDKL model consists of two procedures: meta-learning procedure and adaptation procedure. To make a clear illustration, we introduce frameworks of two procedures and then explain them in detail.

Meta-learning procedure: Learning experience

Our MDKL model uses the kernel in [46] as its base kernel:

$$k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\theta}, \mathbf{p}) = exp(-\sum_{k=1}^{d} \theta_k |x_k^i - x_k^j|^{p_k}). \tag{5.2}$$

Therefore, the deep kernel will be:

$$k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma}) = exp(-\sum_{k=1}^{d} \theta_k |\phi(x_k^i) - \phi(x_k^j)|^{p_k}), \tag{5.3}$$

where $\boldsymbol{\gamma} = \{\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}, \mathbf{p}\}$ is a set of deep kernel parameters. $\phi, \mathbf{w}$ and $\mathbf{b}$ are neural network and its parameters, as explained in Eq.(5.1). Details about other alternative base kernels are available in [109].

The aim of meta-learning procedure is to learn experience $\boldsymbol{\gamma}^e$ from related tasks $\{T_1, \ldots, T_N\}$, including neural network parameters $\mathbf{w}, \mathbf{b}$ that describes the shared expression of correlation functions and the task-independent base kernel parameters $\boldsymbol{\theta}^e, \mathbf{p}^e$. The pseudo code of meta-learning procedure is given in Algorithm 12.

Ideally, the experience $\boldsymbol{\gamma}^e$ is learned from plenty of (N_m) small datasets D_m collected from different related tasks. However, in practice, the number of available related tasks N can be much smaller than N_m. Hence, the meta-learning is conducted gradually over U update iterations (line 2). During each update iteration, a small batch of related tasks contribute B small datasets $\{D_{m1}, \ldots, D_{mB}\}$ for meta-learning purpose (lines 4 and 6). Note that if $N < N_m$, a related task T_i can be used multiple times in the meta-learning procedure.

For a given dataset D_{mi}, we denote $\boldsymbol{\theta}^i = \boldsymbol{\theta}^e + \Delta\boldsymbol{\theta}^i$ and $\mathbf{p}^i = \mathbf{p}^e + \Delta\mathbf{p}^i$ as the task-specific kernel parameters, where $\Delta\boldsymbol{\theta}^i, \Delta\mathbf{p}^i$ are the distance we need to move from the task-independent parameters to the task-specific parameters (line 8). The loss function L of MDKL is the likelihood function defined as follows [46]:

$$\frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}|\mathbf{R}|^{1/2}} exp[-\frac{(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\sigma^2}], \tag{5.4}$$

Algorithm 12 Meta-learning($D_i, N_m, |D_m|, B, \alpha$)

Input:

D_i: Datasets collected from related tasks T_i, i=$\{1, \ldots, N\}$;
N_m: Number of subsets D_m for meta-learning;
$|D_m|$: Size of subsets D_m. $|D_m| \leq |D_i|$ due to $D_m \subseteq D_i$.
B: Number of related tasks in a batch;
α: Learning rate for priors.

Procedure:

1: Randomly initialise $\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^e, \mathbf{p}^e$.
2: Set the number of update iterations U = N_m/B.
3: **for** $j = 1$ to U **do**
4: $\{D'_1, \ldots, D'_B\} \leftarrow$ Randomly select a batch of datasets from $\{D_1, \ldots, D_N\}$.
5: **for all** D'_i in the batch **do**
6: $D_{mi} \leftarrow$ Randomly sample a subset of size $|D_m|$ from D'_i.
7: Initialise task-specific increment $\Delta\boldsymbol{\theta}^i, \Delta\mathbf{p}^i$.
8: Compute task-specific parameters: $\boldsymbol{\theta}^i = \boldsymbol{\theta}^e + \Delta\boldsymbol{\theta}^i$, $\mathbf{p}^i = \mathbf{p}^e + \Delta\mathbf{p}^i$.
9: Obtain deep kernel $k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma})$ based GP: $h(\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$=$\{\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^i, \mathbf{p}^i\}$ (Eq.(5.3)).
10: Compute the loss function $L(D_{mi}, h(\boldsymbol{\gamma}))$ (Eq.(5.4)).
11: **end for**
12: Update $\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^e, \mathbf{p}^e$ using gradient descent: $\alpha \bigtriangledown L(D_{mi}, h(\boldsymbol{\gamma}))$ (Eq.(5.5)).
13: **end for**

Output: Task-independent parameters: $\boldsymbol{\gamma}^e = \{\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^e, \mathbf{p}^e\}$.

where $|\mathbf{R}|$ is the determinant of the correlation matrix $\mathbf{R}$, each element in the matrix is computed through Eq.(5.3). $\mathbf{y}$ is the fitness vector of D_{mi}. μ and σ^2 are the mean and the variance of the prior distribution, respectively. Experience $\boldsymbol{\gamma}^e = \{\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^e, \mathbf{p}^e\}$ is updated by gradient descent (line 12), take $\boldsymbol{\theta}^e$ as an example:

$$\boldsymbol{\theta}^e \leftarrow \boldsymbol{\theta}^e - \frac{\alpha}{B} \sum_{i=1}^{B} \bigtriangledown_{\boldsymbol{\theta}^e} L(D_{mi}, h(\boldsymbol{\gamma})). \tag{5.5}$$

After U iterations, $\boldsymbol{\gamma}^e$ has been trained sufficiently ($\boldsymbol{\gamma}^e$ values do not change anymore) by N_m small datasets D_m and will be used in the evolutionary optimisation of target task T_* later.

Adaptation procedure: Using experience

Algorithm 13 Adaptation($\boldsymbol{\gamma}^*, S_*, \beta$)

Input:

$\boldsymbol{\gamma}^*$: Current surrogate parameters;
S_*: A dataset sampled from target task T_* (Archive).
β: Learning rate for adaptation.

Procedure:

1: **if** $\boldsymbol{\gamma}^* == \boldsymbol{\gamma}^e$ **then**
2: Initialise task-specific increments $\Delta\boldsymbol{\theta}^*, \Delta\mathbf{p}^*$.
3: Compute task-specific parameters: $\boldsymbol{\theta}^* = \boldsymbol{\theta}^e + \Delta\boldsymbol{\theta}^*$, $\mathbf{p}^* = \mathbf{p}^e + \Delta\mathbf{p}^*$.
4: Obtain deep kernel $k(\mathbf{x}^i, \mathbf{x}^j|\boldsymbol{\gamma}^*)$ based GP: $h(\boldsymbol{\gamma}^*)$, where $\boldsymbol{\gamma}^* = \{\mathbf{w}, \mathbf{b}, \boldsymbol{\theta}^*, \mathbf{p}^*\}$ (Eq.(5.3)).
5: **end if**
6: Compute the loss function $L(S_*, h(\boldsymbol{\gamma}^*))$ (Eq.(5.4)).
7: Update $\Delta\boldsymbol{\theta}^*, \Delta\mathbf{p}^*$ using gradient descent: $\beta \bigtriangledown L(S_*, h(\boldsymbol{\gamma}^*))$ (Eq.(5.6)).

Output: Adapted MDKL $h(\boldsymbol{\gamma}^*)$.

The meta-learning of experience $\boldsymbol{\gamma}^e$ enables MDKL to handle a family of related tasks in general. To approximate a specific task T_* well, surrogate $h(\boldsymbol{\gamma}^e)$ needs to adapt task-specific increments $\Delta\boldsymbol{\theta}^*$ and $\Delta\mathbf{p}^*$ in the way described in Algorithm 13. There are three major differences between the meta-learning procedure and the adaptation procedure:

- First, the dataset S_* for adaptations is sampled from target tasks T_*. By comparison, in the meta-learning procedure, the training datasets $\{D_{m1}, \ldots, D_{mN_m}\}$ are sampled from N different related tasks.

- Second, the parameters to be updated are increments $\Delta\boldsymbol{\theta}^*$ and $\Delta\mathbf{p}^*$, not task-independent parameters $\boldsymbol{\theta}^e$ and $\mathbf{p}^e$. Thus, the gradient descent in Eq.(5.5) is replaced by the following one (line 7):

$$\Delta\boldsymbol{\theta}^* \leftarrow \Delta\boldsymbol{\theta}^* - \beta \bigtriangledown_{\Delta\boldsymbol{\theta}^*} L(S_*, h(\boldsymbol{\gamma}^*)). \tag{5.6}$$

- Third, task-specific increments $\Delta\boldsymbol{\theta}^*$ and $\Delta\mathbf{p}^*$ are initialised only when it is the first time to adapt task-specific parameters (lines 1-5). From the perspective of tuning parameters, we can say task-independent base kernel parameters $\boldsymbol{\theta}^e$ and $\mathbf{p}^e$ are the initialisation points for further adaptations. In other words, the learning of a new task starts on the basis of the learned experience.

A diagram of the deep kernel implemented in our MDKL model is illustrated in Fig. 5.2.

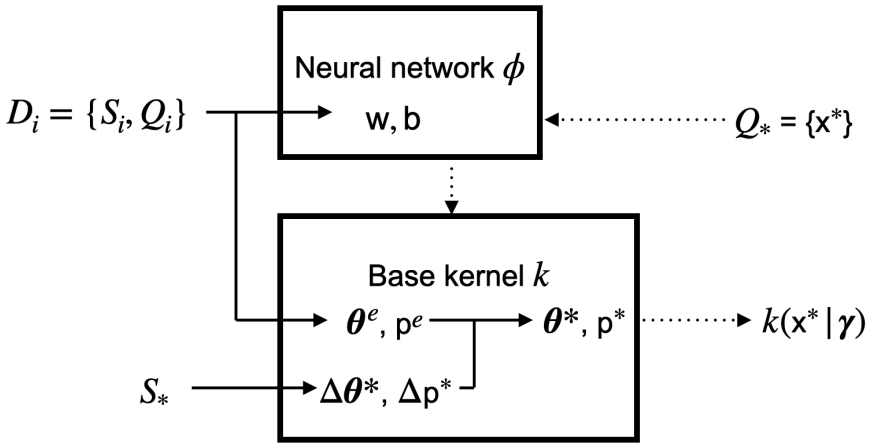


Figure 5.2: Diagram of our deep kernel implementation. The solid lines depict the training process, the dotted lines depict the inference process. Q_* denotes query samples to be evaluated on our surrogates. The neural network ensures the expressive power of our deep kernel. Common features of related tasks are represented by neural network parameters $\mathbf{w}, \mathbf{b}$ and base kernel parameters $\boldsymbol{\theta}^e, \mathbf{p}^e$, which are the learned experience. Task-specific increments $\Delta\boldsymbol{\theta}^*$ and $\Delta\mathbf{p}^*$ distinguishes a given task T_* from other tasks

5.3.3 Surrogate Update Strategy

The adaptation procedure details how experience is used to adapt a MDKL surrogate with a dataset S_* from the target task T_*. In this subsection, we describe the update strategy in our experience-based SAEA framework. To properly integrate experience and data from T_*, our update strategy is designed to determine whether a MDKL surrogate should be adapted in the current iteration or not, ensuring an optimal update frequency of surrogates.

As illustrated in Algorithm 14, the surrogate update starts when a new optimal solution(s) has been evaluated on expensive functions and an updated archive S_* is available. For a given surrogate $h(\boldsymbol{\gamma}^*)$, its mean squared error (MSE) on S_* is chosen as the update criterion: If the MSE after an adaptation e_1 (line 3) is larger than the MSE without an adaptation e_0 (line 1), then the surrogate will roll back to the status before the adaptation. This indicates the surrogate update has been refused and $h(\boldsymbol{\gamma}^*)$ will not be adapted in the

Algorithm 14 Update($\boldsymbol{\gamma}^*, S_*, \beta$)

Input:

$\boldsymbol{\gamma}^*$: Current surrogate parameters;
S_*: Updated archive.
β: Learning rate for further adaptations.

Procedure:

1: $e_0 \leftarrow \text{MSE}(h(\boldsymbol{\gamma}^*), S_*)$.
2: $h(\boldsymbol{\gamma}') \leftarrow \text{Adaptation}(\boldsymbol{\gamma}^*, S_*, \beta)$./*Temporary surrogate, Algorithm 13.*/
3: $e_1 \leftarrow \text{MSE}(h(\boldsymbol{\gamma}'), S_*)$.
4: **if** $e_0 > e_1$ **then**
5: update $\boldsymbol{\gamma}^* = \boldsymbol{\gamma}'$, obtain new $h(\boldsymbol{\gamma}^*)$.
6: **end if**

Output: Surrogate $h(\boldsymbol{\gamma}^*)$.

current iteration. Otherwise, the adapted surrogate will be chosen (line 5). Note that no matter whether surrogate adaptations are accepted or refused, the resulting surrogates will be treated as updated surrogates, which are employed to assist a SAEA optimiser in the next iteration.

5.3.4 Framework Compatibility and Surrogate Usage

Our experience-based SAEA framework is compatible with regression-based SAEAs since the original surrogates in these SAEAs can be replaced by our MDKL surrogates directly. Due to the nature of a GP, when predicting the fitness of a query solution $\mathbf{x}^*$, a MDKL surrogate produces a predictive Gaussian distribution $\mathcal{N}(\hat{y}(\mathbf{x}^*), \hat{s}^2(\mathbf{x}^*))$, the predicted mean $\hat{y}(\mathbf{x}^*)$ and covariance $\hat{s}^2(\mathbf{x}^*)$ are specified as [46]:

$$\hat{y}(\mathbf{x}^*) = \mu + \mathbf{r}'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\mu) \tag{5.7}$$

$$\hat{s}^2(\mathbf{x}^*) = \sigma^2(1 - \mathbf{r}'\mathbf{R}^{-1}\mathbf{r}) \tag{5.8}$$

where $\mathbf{r}$ is a correlation vector consisting of covariances between $\mathbf{x}^*$ and S_*, other variables are explained in Eq.(5.4).

Classification-based SAEAs are not compatible with our SAEA framework. The classification surrogates in these SAEAs are employed to learn the relation between pairs of solutions, or the relation between solutions and a set of reference solutions. The class labels used for surrogate training can be fluctuating during the optimisation and thus hard to be learned over related tasks. Similarly, in ordinal-regression-based SAEAs, the ordinal relation values to be learned are not as stable as the fitness of expensive functions. So ordinal-regression-based SAEAs are also not compatible with our SAEA framework. In this chapter, we focus on experience-based optimisation for regression-based SAEAs, while other SAEA categories are left to be discussed in future work.

5.4 Experimental Studies

Our computational studies can be divided into three parts:

- Section 5.4.1 evaluates the effectiveness of learning experience through a sinusoid function regression problem and a real-world engine modelling problem.
- Sections 5.4.2 to 5.4.5 use the scenario of multi-objective optimisation as an example to investigate the performance of our SAEA framework in depth. Empirical evidence is provided to guide the application of our SAEA framework.
- Section 5.4.6 investigates the performance of our SAEA framework in breadth. The engine calibration problem to be solved in the experiment covers many representative features, such as constrained optimisation, single-objective optimisation, and real-world applications.

For all meta-learning methods used in our experiments, their basic setups are listed in Table 5.1. The neural network structure is suggested by [23, 69], and the learning rates are the default values that have been widely used in many meta-learning methods [23, 32, 69].

Table 5.1: Parameter setups for meta-learning methods.

Module	Parameter	Value
Meta-learning	Number of meta-learning datasets N_m	20000
	Number of update iterations U	2000
	Batch size B	10
Neural network	Number of hidden layers	2
	Number of units in each hidden layer	40
	Learning rates α, β	0.001, 0.001
	Activation function	ReLU

5.4.1 Effectiveness of Learning Experience

The evaluation of the effectiveness of learning experience aims at demonstrating that our MDKL model is able to learn experience from related tasks and it outperforms other meta-learning models. For this reason, the experiment is designed to answer the following questions:

- Given a small dataset S_* from target task T_*, can MDKL learn experience from related tasks and then generate a model that has the smallest MSE?
- If yes, which components of MDKL contribute to the effectiveness of learning experience? Meta-learning or/and deep kernel learning? If not, why not?

To answer the two questions above, we consider two experiments to evaluate the effectiveness of learning experience: amplitude prediction for unknown periodic sinusoid functions, and fuel consumption prediction for a gasoline motor engine. The former is a few-shot regression problem that motivates many meta-learning studies [23, 32, 101, 69], while the latter is a real-world regression problem [123].

A. Sinusoid Function Regression

In the sinusoid regression experiment, we learn experience from a series of 1-dimensional sinusoid functions:

$$y = Asin(wx + b) + \epsilon \tag{5.9}$$

where the amplitude A and phase w of sine waves are varied between functions. The target is to approximate an unknown sinusoid function with a small support dataset S_* and the learned experience. Clearly, by integrating experience with S_*, we estimate parameters (A, w, b) for an unknown sinusoid function. As a result, the output y of the given sinusoid function can be predicted once a query data x is inputted.

Generation of Sinusoid Function Variants:

As suggested in [23, 32], we set amplitude $A \in$ [0.1, 5.0], frequency $w \in [0.999, 1.0]$, phase $b \in [0, \pi]$, and Gaussian noise $\epsilon \sim (0, 0.1)$. Therefore, a sinusoid function can be generated by sampling three parameters (A, w, b) from their ranges uniformly. In total, $N_m = N =$ 20000 related sinusoid functions are generated at random.

Experimental Setups:

All data points x are sampled from the range $\in$ [-5.0, 5.0]. In the meta-learning procedure, both support set and query set contain 5 data points. Hence, a dataset D_i is sampled from each (related) sinusoid function T_i, and $|D_i| = |D_m| = 10$. Six experiments are conducted where $|S_*| = \{2, 3, 5, 10, 20, 30\}$ data points are sampled from the target function. Considering Gaussian noise ϵ could be relatively large when amplitude A is close to 0.1, normalised mean squared error (NMSE) is chosen as a performance indicator. NMSE is measured using a dataset that contains 100 data points sampled uniformly from the x range.

Comparison methods:

In this experiment, three families of modelling methods are compared with our MDKL model:

- **Meta-learning modelling methods** that were proposed for regression tasks: MAML [23], ALPaCA [32], and DKT [69]. The configurations of MAML, ALPaCA, and DKT are the same as suggested in the original literature.

- **Non-meta-learning modelling method** that is widely used for regression tasks: the GP model (also known as Kriging model [91] or the design and analysis of computer experiments (DACE) stochastic process model [83]). We choose a GP as a baseline since it is effective and more relevant to MDKL than other non-meta-learning modelling methods. We set the range of base kernel parameters in the GP model as $\theta \in [10^{-5}, 10]$ and $p \in [1, 2]$.

- **MDKL related methods** that are designed to investigate which components of MDKL contribute to the modelling performance: GP_Adam, DKL, and MDKL_NN. GP_Adam is a GP model fitted by Adam optimiser. The combination of GP_Adam and a neural network results in a kind of DKL algorithm. MDKL_NN is a meta-learning version of DKL, but it learns only neural network parameters through meta-learning and has no task-independent base kernel parameters.

Results and Analysis:

Table 5.2 reports the statistical test results of the NMSE values achieved by comparison algorithms in sinusoid function regression experiments. Each row lists the results obtained when the same number of fitness evaluations $|S_*|$ are used to train models. The results of Wilcoxon rank sum test between MDKL and other compared algorithms are listed in the last row. It can be observed that both MDKL and DKT have achieved the smallest NMSE values in all tests in the comparison with other meta-learning and non-meta-learning modelling methods.

Additional Wilcoxon rank sum tests have been made between MDKL related algorithms to answer our second question (not reported in Table 5.2). All test results are obtained under the same number of fitness evaluations $|S_*|$. The statistical test results between DKL and GP_Adam are 5/1/0, showing that DKL is preferable to GP_Adam when only a few data points are available for modelling. Hence, using a neural network to build a deep kernel

Table 5.2: Mean Normalised Mean Squared Error (NMSE) and standard deviation (in brackets) of 30 runs on the amplitude regression of sinusoid function. GP [91] is a widely used surrogate in SAEAs, MAML [23], ALPaCA [32], and DKT [69] are meta-learning methods. GP_Adam is a GP model fitted by Adam optimiser. DKL is a deep kernel learning algorithm that adds a neural network to GP_Adam. MDKL_NN applies meta-learning to DKL, but no task-independent base kernel parameters are shared between related tasks. Support data points are used to train non-meta surrogates or adapt meta-learning surrogates. '+', '≈', and '−' denote MDKL is statistically significantly superior to, almost equivalent to, and inferior to the compared modelling methods in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. It shows that MDKL and DKT have lower NMSE than other models. The effectiveness of meta-learning on both the neural network and the base kernel has been demonstrated on this example.

Support data points $\|S_*\|$	GP [91]	GP_Adam	DKL	MDKL_NN	MDKL	DKT [69]	MAML [23]	ALPaCA [32]
2	1.63e-1(9.18e-2)≈	1.93e-1(9.72e-2)+	1.63e-1(9.05e-2)≈	1.57e-1(9.26e-2)≈	1.56e-1(9.49e-2)	1.56e-1(9.49e-2)≈	2.09e-1(3.63e-1)≈	1.07e+0(2.57e+0)≈
3	1.27e-1(6.04e-2)≈	1.62e-1(6.53e-2)+	1.21e-1(5.96e-2)≈	1.16e-1(5.95e-2)≈	1.10e-1(6.20e-2)	1.10e-1(6.20e-2)≈	2.09e-1(3.60e-1)≈	4.36e-1(8.57e-1)≈
5	6.76e-2(4.62e-2)≈	1.09e-1(5.61e-2)+	7.52e-2(4.40e-2)+	6.38e-2(3.91e-2)≈	4.79e-2(3.73e-2)	4.79e-2(3.70e-2)≈	2.08e-1(3.59e-1)+	4.31e-1(8.04e-1)≈
10	1.70e-2(1.87e-2)≈	6.13e-2(4.58e-2)+	2.87e-2(1.89e-2)+	1.89e-2(1.61e-2)+	1.07e-2(1.16e-2)	1.09e-2(1.17e-2)≈	2.08e-1(3.58e-1)+	6.59e-1(2.14e+0)+
20	5.42e-3(7.64e-3)+	3.92e-2(4.29e-2)+	9.64e-3(1.02e-2)+	5.24e-3(6.57e-3)+	2.57e-3(4.53e-3)	2.63e-3(4.61e-3)≈	2.08e-1(3.58e-1)+	1.13e-1(3.39e-1)+
30	3.97e-3(7.40e-3)+	3.32e-2(4.18e-2)+	4.81e-3(6.68e-3)+	3.20e-3(5.85e-3)+	1.68e-3(3.61e-3)	1.60e-3(3.39e-3)≈	2.08e-1(3.58e-1)+	7.59e-2(2.01e-1)+
$+/\approx/-$	2/4/0	6/0/0	4/2/0	3/3/0	-/-/-	0/6/0	4/2/0	3/3/0

for GP is able to enhance the performance of modelling. When meta-learning technique is applied to DKL, the statistical test results between MDKL_NN and DKL are 3/3/0. The meta-learning of neural network parameters is necessary since it contributes to the performance of MDKL. Further statistical test between MDKL and MDKL_NN gives results of 3/3/0, indicating that the meta-learning of base kernel parameters is effective on this regression problem.

B. Estimation of Engine Fuel Consumption

We then consider a BSFC regression task for a gasoline motor engine [123], where BSFC is evaluated on a gasoline engine simulation (denoted by T_* in this engine experiment) provided by Ford Motor Company.

Experimental setups:

The related tasks T_i used in our experiment are $N = 100$ gasoline engine models. These engine models have different behaviours when compared with T_*, but they share the basic

Table 5.3: Mean Mean Squared Error (MSE) and standard deviation (in brackets) of 30 runs on the regression of engine fuel consumption. GP [91] is a widely used surrogate in SAEAs, MAML [23] and ALPaCA [32] are meta-learning methods. GP_Adam is a GP model fitted by Adam optimiser. DKL is a deep kernel learning algorithm that adds a neural network to GP_Adam. MDKL_NN applies meta-learning on DKL, but no task-independent base kernel parameters are shared between related tasks. Support data points are used to train non-meta surrogates or adapt meta-learning surrogates. All results are normalised since the actual engine data is unable to be disclosed. '+', '≈', and '−' denote MDKL is statistically significantly superior to, almost equivalent to, and inferior to the compared modelling methods in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. It shows that MDKL and MDKL_NN have achieved the smallest MSE. The effectiveness of meta-learning on the neural network has been demonstrated on this example.

Support data points $\|S_*\|$	GP [91]	GP_Adam	DKL	MDKL_NN	MDKL	DKT [69]	MAML [23]	ALPaCA [32]
2	2.23e+1(3.20e+0)+	2.37e+1(6.30e+0)+	2.30e+1(5.87e+0)+	1.73e+1(6.33e+0)≈	1.72e+1(6.34e+0)	1.81e+1(5.68e+0)≈	1.87e+1(6.37e+0)≈	1.91e+1(1.02e+1)≈
3	2.14e+1(3.74e+0)+	2.41e+1(1.38e+1)+	2.20e+1(3.74e+0)+	1.45e+1(7.13e+0)≈	1.45e+0(7.01e+0)	1.55e+1(6.66e+0)≈	1.80e+1(4.69e+0)≈	2.13e+1(1.97e+1)≈
5	2.13e+1(3.27e+0)+	2.46e+1(1.00e+1)+	2.07e+1(3.95e+0)+	1.12e+1(6.65e+0)≈	1.10e+1(6.58e+0)	1.21e+1(6.49e+0)≈	1.84e+1(6.05e+0)+	1.99e+1(2.29e+1)+
10	1.84e+1(1.89e+0)+	2.06e+1(1.19e+1)+	2.10e+1(5.79e+0)+	7.19e+0(4.82e+0)≈	7.08e+0(4.77e+0)	7.99e+0(4.87e+0)≈	1.70e+1(5.54e+0)+	1.38e+1(8.12e+0)+
20	1.56e+1(2.00e+0)+	2.38e+1(1.05e+1)+	1.76e+1(2.42e+0)+	5.03e+0(1.82e+0)≈	4.86e+0(1.71e+0)	5.74e+0(1.91e+0)+	1.50e+1(2.59e+0)+	1.01e+1(5.52e+0)+
40	1.28e+1(2.03e+0)+	1.48e+1(7.35e+0)+	1.67e+1(3.73e+0)+	4.13e+0(7.90e-1)≈	4.00e+0(8.59e-1)	4.92e+0(1.09e+0)+	1.45e+1(1.85e+0)+	8.01e+0(3.35e+0)+
+/≈/−	6/0/0	6/0/0	6/0/0	0/6/0	-/-/-	2/4/0	4/2/0	4/2/0

features of gasoline engines. All related tasks and the target task have the same six decision variables. Each related task T_i provides only 60 solutions, forming a dataset D_i. The size of datasets used for meta-learning $|D_m|$ is set to 40. Six tests are conducted where $|S_*| = \{2, 3, 5, 10, 20, 40\}$ data points are sampled from the real engine simulation T_*. MSE is chosen as an indicator of modelling accuracy, which is measured using a dataset consisting of 12500 data points that are sampled uniformly from the engine decision space.

Comparison methods:

The comparison algorithms are the same as described in Section 5.4.1.A.

Results and analysis:

The statistical test results of the MSE values achieved by comparison algorithms on BSFC regression are summarised in Table 5.3. Each row lists the results obtained when the same number of fitness evaluations $|S_*|$ are used to train models. The result of Wilcoxon rank sum test between MDKL and other compared algorithms is listed in the last row. It can

be observed that MDKL and MDKL_NN outperform other comparison modelling methods since they have achieved the smallest MSE in all tests. The MDKL model learned from 20 data points on the target problem is still competitive to the compared models that are learned from 40 data points on the target problem, showing the effectiveness of our learning experience.

Contributions of MDKL components are analyzed through statistical tests between MDKL related methods. Again, all test results are obtained under the same number of fitness evaluations $|S_*|$. The statistical test results between DKL and GP_Adam are 1/5/0, indicating that the neural network in DKL makes some contributions to the performance of MDKL. The statistical test results between MDKL_NN and DKL are 6/0/0, demonstrating that the meta-learning of neural network parameters constructs a useful deep kernel and contributes to the improvement of modelling accuracy. However, there is no significant difference between the performance of MDKL and that of MDKL_NN, the meta-learning on base kernel parameters does not play a critical role on this engine problem. In comparison, the meta-learning on base kernel parameters is effective in sinusoid function regression experiments. Besides, the statistical test results between MDKL_NN and MAML are 4/2/0. Considering that MAML is a neural network regressor learned through meta-learning, we can conclude that GP is an essential component of our MDKL. In summary, all components in MDKL are necessary, they all contribute to the effectiveness of learning experience.

C. Discussion

The comparison experiments on the gasoline motor engine and sinusoid functions have demonstrated the effectiveness of our MDKL modelling method in the learning of experience for these problems. Given a small dataset of the target task, the model learned through MDKL method has the smallest MSE or NMSE among all comparison models. Besides, the investigation between MDKL and its variants shows that all components in MDKL have made their contributions to the effectiveness of learning experience. However, similar to

other meta-learning studies [23, 32], we have not defined the similarity between tasks. In other words, the boundary between related tasks and unrelated tasks has not been defined. This should be a topic of further study on meta-learning. Additionally, the relationship between task similarity and modelling performance has not been investigated. Instead of it, we study the relationship between task similarity and SAEA optimisation performance in Section 5.4.4, since our main focus is the surrogate-assisted evolutionary optimisation.

5.4.2 Performance on Expensive Multi-Objective Optimisation

So far we have shown the effectiveness of our experience learning method. In the following subsections, we aim to demonstrate the effectiveness of our experience-based SAEA framework. The experiment in this subsection is designed to answer the question below:

- With the experience learned from related tasks, can our SAEA framework helps a SAEA to save $9d$ solutions without a loss of optimisation performance?

The computational study is conducted on DTLZ benchmark test problems [18]. All DTLZ problems in our work have $d = 10$ decision variables and 3 objectives, as the setups that have been widely used in [68, 90].

Generation of DTLZ variants:

The details of generating DTLZ variants (related tasks) are given as follows. The DTLZ optimisation experiment generates m-objective DTLZ variants in the following ways:

DTLZ1:

$$f_1 = (a_1 + g)0.5 \prod_{i=1}^{m-1} x_i, \tag{5.10}$$

$$f_{j=2:m-1} = (a_j + g)(0.5 \prod_{i=1}^{m-j} x_i)(1 - x_{m-j+1}), \tag{5.11}$$

$$f_m = (a_m + g)0.5(1 - x_1), \tag{5.12}$$

$$g = 100\left[k + \sum_{i=1}^{k}\left((z_i - 0.5)^2 - cos\left(20\pi(z_i - 0.5)\right)\right)\right], \tag{5.13}$$

where $\mathbf{z}$ is a vector consisting of the last $k = d - m + 1$ variables in $\mathbf{x}$. In other words, $\mathbf{z} = \{z_1, \ldots, z_k\} = \{x_m, \ldots, x_d\}$. The variants of DTLZ1 introduce only one variable $\mathbf{a} \in [0.1, 5.0]^m$ in Eq.(5.10), Eq.(5.11), and Eq.(5.12), where $\mathbf{a} = \mathbf{1}$ in the original DTLZ1. For out-of-range test, $\mathbf{a} \in [1.5, 5.0]^m$.

DTLZ2:

$$f_1 = (a_1 + g)\prod_{i=1}^{m-1} cos(\frac{x_i\pi}{b_1}), \tag{5.14}$$

$$f_{j=2:m-1} = (a_j + g)\left(\prod_{i=1}^{m-j} cos(\frac{x_i\pi}{b_j})\right) sin(\frac{x_{m-j+1}\pi}{b_j}), \tag{5.15}$$

$$f_m = (a_m + g)sin(\frac{x_1\pi}{b_m}), \tag{5.16}$$

$$g = \sum_{i=1}^{k}(z_i - 0.5)^2. \tag{5.17}$$

The variants of DTLZ2 introduce two variables $\mathbf{a} \in [0.1, 5.0]^m$ and $\mathbf{b} \in [0.5, 2.0]^m$ in Eq.(5.14), Eq.(5.15), and Eq.(5.16), where $\mathbf{a} = \mathbf{1}$ and $\mathbf{b} = \mathbf{2}$ in the original DTLZ2. For out-of-range test, $\mathbf{a} \in [1.5, 5.0]^m, \mathbf{b} \in [0.5, 1.5]^m$.

DTLZ3: The variants of DTLZ3 are generated using the same way as described in DTLZ2, except the equation g from Eq.(5.17) is replaced by the one from Eq.(5.13).

DTLZ4: The variants of DTLZ4 are generated using the same way as described in DTLZ2, except all x_i are replaced by x_i^{100}.

DTLZ5: The variants of DTLZ5 are generated using the same way as described in DTLZ2, except all $x_2, \ldots, x_{m-1}$ are replaced by $\frac{1+2gx_i}{2(1+g)}$.

DTLZ6:

$$g = \sum_{i=1}^{k} z_i^{0.1}. \tag{5.18}$$

The variants of DTLZ6 are generated using the same way as described in DTLZ5, except

the equation g from Eq.(5.17) is replaced by the one from Eq.(5.18).

DTLZ7:

$$f_{j=1:m-1} = x_j + a_j, \tag{5.19}$$

$$f_m = (1+g)\left(m - \sum_{i=1}^{m-1}\left[\frac{f_i}{1+g}\left(1+sin(3\pi f_i)\right)\right]\right), \tag{5.20}$$

$$g = a_m + 9\sum_{i=1}^{k}\frac{z_i}{k}. \tag{5.21}$$

The variants of DTLZ7 introduce one variable $\mathbf{a} \in [0.1, 5.0]^m$ in Eq.(5.19) and Eq.(5.21), where $a_{j=1:m-1} = 0$ and $a_m = 1$ in the original DTLZ7. For out-of-range test, $\mathbf{a} \in [1.5, 5.0]^m$.

Comparison algorithms:

As we explained in Section 5.3.4, our experience-based SAEA framework is compatible with regression-based SAEAs. Hence, we select MOEA/D-EGO [120] as an example and replace its GP surrogates by our MDKL surrogates. The resulting algorithm is denoted as MOEA/D-EGO(EB). Note that it is not necessary to specially select a newly proposed regression-based SAEA as our example, our main objective is to save evaluations with experience and observe if there is any damage to the optimisation performance caused by the saving of evaluations. Therefore, it does not make any difference which regression-based SAEA we choose as our example. Additionally, to demonstrate the improvement of optimisation performance caused by using experience on DTLZ problems is significant, several state-of-the-art SAEAs are also compared as baselines, including ParEGO [47], K-RVEA [8], CSEA [68], OREA [115], and KTA2 [90]. Among these SAEAs, ParEGO, K-RVEA, and KTA2 use regression-based surrogates, CSEA uses a classification-based surrogate, and OREA employs an ordinal-regression-based surrogate. Note that all comparison SAEAs are not capable of learning and using experience from related tasks. Hence, for comparison SAEAs, using the data from related tasks to train their surrogates will make no difference to their optimisation performance.

Table 5.4: Parameter setups for DTLZ optimisation.

Parameter	MOEA/D-EGO(EB)	Comparisons
Number of related tasks N	20000 (N_m in Table 5.1)	-
Size of datasets from related tasks $\|D_i\|$	20 ($2d$)	-
Size of datasets for meta-learning $\|D_m\|$	$\|D_i\|$	-
Evaluations for initialisation	10 ($1d$)	100 ($10d$)
Evaluations for further optimisation	50	50
Total evaluations	60	150

We implement the experience-based SAEA framework, MOEA/D-EGO, ParEGO, and OREA, while the code of K-RVEA, CSEA, and KTA2 is available on PlatEMO [99], an open source MATLAB platform for evolutionary multi-objective optimisation. To make a fair comparison, all comparison algorithms share the same initial dataset S_* in an independent run. We also set $\boldsymbol{\theta} \in [10^{-5}, 100]^d$ and $\mathbf{p} = \mathbf{2}$ for all GP surrogates as suggested in [90], these GP surrogates are implemented through DACE [83]. Other configurations are the same as suggested in their original literature.

Experimental setups:

The parameter setups for this multi-objective optimisation experiment are listed in Table 5.4. In the meta-learning procedure, we assume plenty of DTLZ variants are available, thus $N = N_m = 20000$, each DTLZ variant T_i provides $|D_i| = |D_m| = 20$ samples for learning experience. During the optimisation process, an initial dataset S_* is sampled using Latin-Hypercube Sampling (LHS) method [60], then extra evaluations are conducted until the evaluation budget has run out. Please note that our purpose is using related tasks to save $9d$ evaluations without a loss of SAEA optimisation performance. Hence, the total evaluation budgets for MOEA/D-EGO(EB) and comparison algorithms are different.

Since the test problems have 3 objectives, we employ inverted generational distance plus (IGD+) [38] as our performance indicator, where smaller IGD+ values indicate better optimisation results. 5000 reference points are generated for computing IGD+ values, as suggested in [68].

Table 5.5: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 runs on the optimisation of DTLZ test problems. MOEA/D-EGO(EB) replaces the original GP [91] surrogates in MOEA/D-EGO [120] with our MDKL surrogates. MOEA/D-EGO(EB) and comparison algorithms initialise their surrogates with 10, 100 samples, respectively. Extra 50 evaluations are allowed in the further optimisation. '+', '≈', and '−' denote MOEA/D-EGO(EB) is statistically significantly superior to, almost equivalent to, and inferior to the compared algorithms in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. Competitive results can be observed from the comparisons between MOEA/D-EGO(EB) and MOEA/D-EGO, while 90 evaluations are saved using the experience learned from related tasks.

Problem	MOEA/D-EGO	MOEA/D-EGO(EB)	ParEGO	K-RVEA	KTA2	CSEA	OREA
DTLZ1	1.07e+2(2.05e+1)+	9.70e+1(1.87e+1)	7.82e+1(1.54e+1)−	1.18e+2(2.45e+1)+	1.01e+2(2.38e+1)≈	1.10e+2(2.50e+1)+	1.02e+2(1.97e+1)≈
DTLZ2	2.99e-1(7.01e-2)+	1.43e-1(2.29e-2)	3.17e-1(4.12e-2)+	2.69e-1(5.97e-2)+	2.14e-1(3.84e-2)+	2.98e-1(5.25e-2)+	1.76e-1(4.69e-2)+
DTLZ3	3.15e+2 (6.04e+1)+	1.97e+2 (1.64e+1)	2.30e+2 (5.99e+1)≈	3.24e+2 (5.90e+1)+	2.67e+2 (6.70e+1)+	2.82e+2(6.97e+1)+	2.72e+2(6.88e+1)+
DTLZ4	5.04e-1(8.25e-2)≈	4.44e-1(1.35e-1)	5.44e-1(7.58e-2)+	4.57e-1(1.14e-1)≈	4.51e-1(9.54e-2)≈	4.75e-1(1.09e-1)≈	3.18e-1(1.54e-1)−
DTLZ5	2.39e-1(7.17e-2)+	1.13e-1(2.24e-2)	2.58e-1(3.68e-2)+	1.92e-1 (5.97e-2)+	1.44e-1(4.60e-2)+	2.14e-1(4.05e-2)+	7.84e-2(2.42e-2)−
DTLZ6	1.29e+0(4.74e-1)≈	1.11e+0(5.71e-1)	1.67e+0(6.77e-1)+	4.62e+0(6.42e-1)+	3.37e+0(6.71e-1)+	6.26e+0(3.40e-1)+	4.60e+0(1.19e+0)+
DTLZ7	3.31e-1(3.11e-1)−	2.47e+0(1.89e+0)	3.66e-1(1.31e-1)−	1.74e-1(3.57e-2)−	4.34e-1(2.20e-1)−	4.17e+0(1.13e+0)+	2.14e+0(1.15e+0)≈
+/ ≈ /−	4/2/1	-/-/-	4/1/2	5/1/1	4/2/1	6/1/0	3/2/2

Results and analysis:

The statistical test results of IGD+ values achieved by comparison algorithms on DTLZ test problems are reported in Tables 5.5. It can be seen from Table 5.5 that, although 90 fewer evaluations are used in surrogate initialisation, MOEA/D-EGO(EB) can still achieve competitive or even smaller IGD+ values than MOEA/D-EGO on all DTLZ problems except for DTLZ7. Fig. 5.3 also shows that the minimal IGD+ values obtained by MOEA/D-EGO(EB) drop rapidly, especially during the first few evaluations, implying the experience learned from DTLZ variants are effective. Therefore, in most situations, our experience-based SAEA framework is able to assist MOEA/D-EGO in reaching competitive or even better optimisation results, with the number of evaluations used for surrogate initialisation reduced from $10d$ to only $1d$.

MOEA/D-EGO(EB) is less effective on DTLZ7 than on other DTLZ problems, which might be attributed to the discontinuity of Pareto front on DTLZ7. Note that MOEA/D-EGO(EB) learns experience from small datasets such as D_m and S_*. The solutions in these small datasets are sampled at random, hence, the probability of having optimal solutions

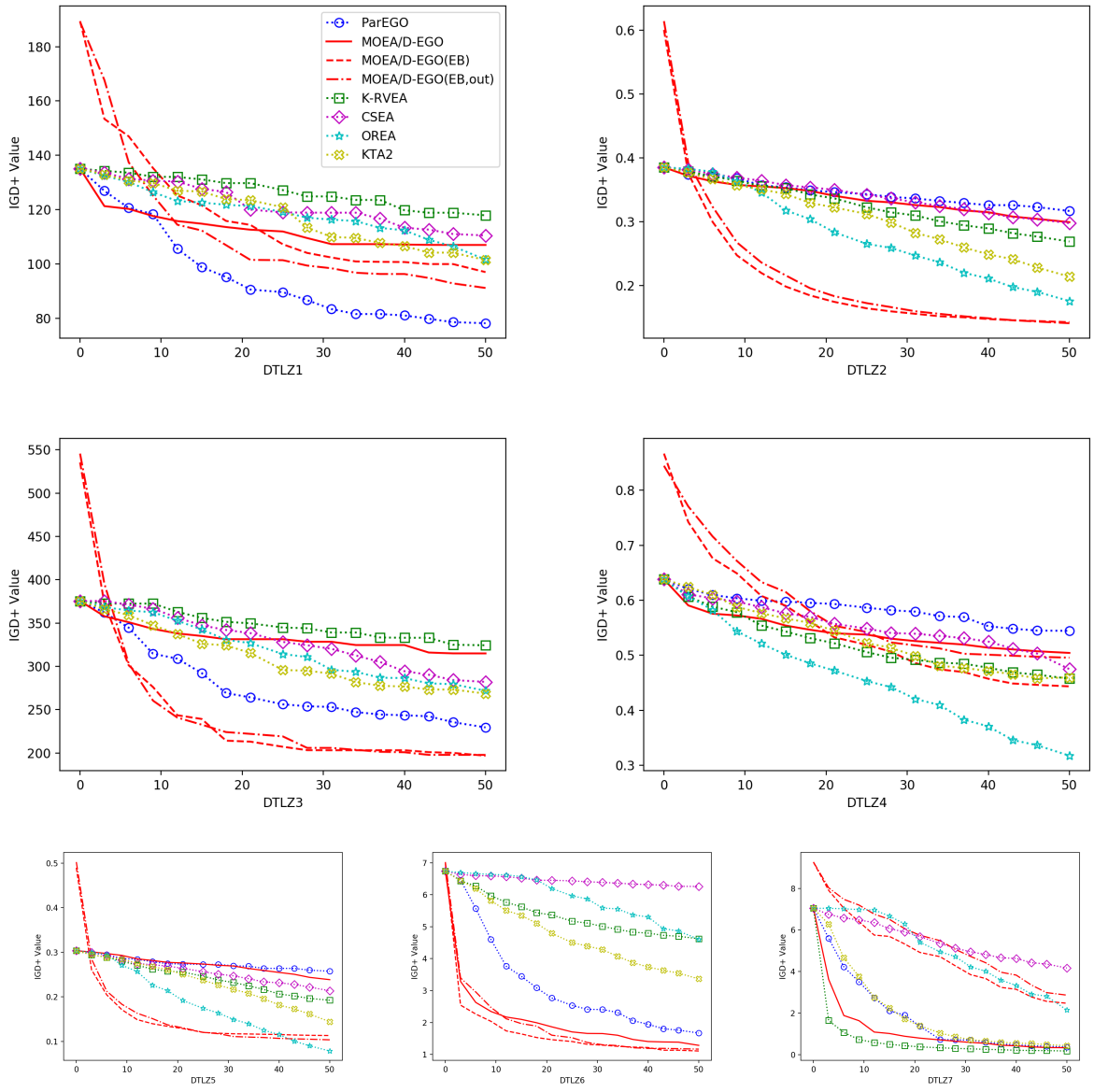


Figure 5.3: Mean Inverted Generational Distance Plus (IGD+) values of 30 runs on the optimisation of DTLZ test problems. MOEA/D-EGO(EB)s replace the original GP [91] surrogates in MOEA/D-EGO [120] with our MDKL surrogates. Note that 'EB out' indicates the target task is excluded from the range of related tasks during the meta-learning procedure. MOEA/D-EGO(EB)s and comparison algorithms initialise their surrogates with 10, 100 samples, respectively. Extra 50 evaluations are allowed in the further optimisation. X-axis denotes the number of evaluations used after the surrogate initialisation. In comparison to MOEA/D-EGO, both MOEA/D-EGO(EB)s achieve smaller IGD+ values on DTLZ1, DTLZ2, DTLZ3, DTLZ5 and competitive IGD+ values on DTLZ4, DTLZ6, while 90 evaluations are saved by using experience.

being sampled is small. However, it is difficult to learn the discontinuity of Pareto front from the sampled non-optimal solutions. As a result, the experience of 'there are four discrete optimal regions' cannot be learned through such small datasets ($|D_m| = 20$) collected from related tasks. However, although MOEA/D-EGO(EB) achieves a larger IGD+ value than MOEA/D-EGO on DTLZ7, the application of our experience-based SAEA framework is still acceptable since the 90 evaluations saved from surrogate initialisation can be used for further evolutionary optimisation.

The experience learned from related tasks also makes MOEA/D-EGO more competitive when compared with other SAEAs. By using MDKL surrogates, significant improvements of optimisation results in terms of the IGD+ value are achieved on DTLZ1, DTLZ2, DTLZ3, and DTLZ5. As a result, MOEA/D-EGO(EB) achieves the smallest IGD+ values on DTLZ2 and DTLZ3, and its optimisation results on DTLZ1 and DTLZ5 are much closer to the best optimisation results (e.g. results obtained by ParEGO and OREA) than MOEA/D-EGO. Although MOEA/D-EGO(EB) does not achieve the smallest IGD+ values on all DTLZ problems, it should be noted that MOEA/D-EGO(EB) is still the best algorithm among comparison SAEAs due to its overall performance. From the statistical test results reported in the last row of Table 5.5, we can observe that no comparison SAEA outperforms MOEA/D-EGO(EB) on three DTLZ problems, but MOEA/D-EGO(EB) outperforms all comparison SAEA on at least three DTLZ problems. Furthermore, the IGD+ values of MOEA/D-EGO(EB) are achieved with an evaluation budget of 60 (10 in surrogate initialisation and 50 in evolutionary optimisation). By comparison, the IGD+ values of other SAEAs are reached with a cost of 150 evaluations, including 100 evaluations used in surrogate initialisation and 50 evaluations in evolutionary optimisation.

The statistical test results reported in the last row of Table 5.5 show that ParEGO and OREA are the best two comparison algorithms when compared with our MOEA/D-EGO(EB). In this paragraph, we want to discuss the advantages of MOEA/D-EGO(EB) when no extra evaluation is saved. For this purpose, we compare the optimisation perfor-

Table 5.6: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 runs on the optimisation of DTLZ test problems. MOEA/D-EGO(EB) is compared with ParEGO and OREA under the same evaluation budget: 10 evaluations for surrogate initialisation and 50 evaluations for the optimisation process. '+', '≈', and '−' denote MOEA/D-EGO(EB) is statistically significantly superior to, almost equivalent to, and inferior to the compared two algorithms in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results.

Problem	MOEA/D-EGO(EB)	ParEGO	OREA
DTLZ1	9.70e+1(1.87e+1)	6.70e+1(4.75e+0)−	1.10e+2(3.65e+1)≈
DTLZ2	1.43e-1(2.29e-2)	5.51e-1(5.37e-2)+	4.28e-1(6.68e-2)+
DTLZ3	1.97e+2 (1.64e+1)	1.84e+2(8.86e+0)≈	2.72e+2(6.59e+1)+
DTLZ4	4.44e-1(1.35e-1)	6.29e-1(7.99e-2)+	6.45e-1(1.24e-1)+
DTLZ5	1.13e-1(2.24e-2)	4.32e-1(8.88e-2)+	3.02e-1(7.63e-2)+
DTLZ6	1.11e+0(5.71e-1)	1.03e+0(4.78e-1)≈	5.71e+0(6.73e-1)+
DTLZ7	2.47e+0(1.89e+0)	4.38e-1(1.39e-1)−	7.12e+0(1.77e+0)+
+/ ≈ /−	-/-/-	3/2/2	6/1/0

mance of these three SAEAs under the same evaluation budget: 10 evaluations ($1d$) for surrogate initialisation and 50 evaluations for further optimisation. The statistical test results are reported in Table 5.6. It can be seen that our MOEA/D-EGO(EB) generally outperforms the compared SAEAs when only $1d$ evaluations are used to initialise their surrogates. The effectiveness of our experience-based SAEA framework has been demonstrated on these examples. Note that OREA is an evolutionary algorithm assisted by ordinal regression surrogates. Currently, our experience-based SAEA framework is applicable to the SAEAs working with fitness regression surrogates. The meta-learning of ordinal regression models can be a topic of further research.

The question raised at the beginning of this subsection can be answered by the results discussed so far. Due to the integration of the experience learned from related tasks (DTLZ variants), although the evaluation cost of surrogates initialisation has been reduced from $10d$ to $1d$, our experience-based SAEA framework is still capable of assisting regression-based SAEAs to reach competitive or even better optimisation results in most situations.

Table 5.7: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 runs on the optimisation of DTLZ test problems. MOEA/D-EGO(EB) replaces the original GP [91] surrogates in MOEA/D-EGO [120] with our MDKL surrogates. MOEA/D-EGO(EB) and comparison algorithms initialise their surrogates with 10, 60 samples, respectively. Extra 30 evaluations are allowed in the further optimisation. '+', '≈', and '−' denote MOEA/D-EGO(EB) is statistically significantly superior to, almost equivalent to, and inferior to the compared algorithms in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. Performance improvement can be observed from the comparisons between MOEA/D-EGO(EB) and MOEA/D-EGO, while 50 evaluations are saved from surrogate initialisation.

Problem	MOEA/D-EGO	MOEA/D-EGO(EB)	ParEGO	K-RVEA	KTA2	CSEA	OREA
DTLZ1	1.07e+2(2.73e+1)≈	1.03e+2(2.34e+1)	8.70e+1(2.53e+1)−	1.22e+2(3.26e+1)+	1.15e+2 (3.18e+1)≈	1.08e+2(2.68e+1)≈	1.11e+2(2.25e+1)+
DTLZ2	3.49e-1(5.82e-2)+	1.57e-1(2.29e-2)	3.51e-1(5.01e-2)+	3.72e-1(4.40e-2)+	3.57e-1(4.68e-2)+	3.55e-1(5.23e-2)+	3.14e-1(3.76e-2)+
DTLZ3	3.07e+2 (5.32e+1)+	2.03e+2(2.42e+1)	2.16e+2(4.89e+1)≈	3.53e+2(7.89e+1)+	3.23e+2(8.82e+1)+	3.35e+2 (6.95e+1)+	3.39e+2(7.72e+1)+
DTLZ4	5.45e-1(1.09e-1)≈	4.91e-1(1.24e-1)	6.36e-1(8.67e-2)+	5.53e-1(9.96e-2)≈	5.47e-1(1.04e-1)≈	5.84e-1(9.75e-2)+	5.14e-1(1.21e-1)≈
DTLZ5	2.79e-1 (5.69e-2)+	1.18e-1(2.25e-2)	2.78e-1(5.59e-2)+	2.82e-1(5.52e-2)+	2.60e-1(5.59e-2)+	2.77e-1(4.41e-2)+	1.99e-1(4.53e-2)+
DTLZ6	2.04e+0(7.33e-1)+	1.29e+0(6.44e-1)	2.47e+0(7.39e-1)+	5.23e+0(6.27e-1)+	4.58e+0(6.47e-1)+	6.44e+0 (3.59e-1)+	5.79e+0(6.70e-1)+
DTLZ7	1.90e+0(9.19e-1)−	4.16e+0(2.54e+0)	1.39e+0(1.49e+0)−	3.13e-1(6.17e-2)−	2.05e+0 (2.20e+0)−	5.47e+0(1.34e+0)+	5.51e+0(1.32e+0)+
+/ ≈ /−	4/2/1	-/-/-	4/1/2	5/1/1	4/2/1	6/1/0	6/1/0

5.4.3 Performance on Extremely Expensive Multi-Objective Optimisation

In this subsection and the next two subsections, we conduct three experiments to investigate the performance of our SAEA framework in-depth. We aim at concluding some empirical guidelines to help the application of our experience-based SAEA framework.

The aim of this subsection is to answer the question below:

- Is our experience-based SAEA framework more suitable for the optimisation problems in which evaluations are extremely expensive? In other words, will the advantage of our SAEA framework become more prominent if the optimisation problems allow a smaller evaluation budget?

We conduct the experiment described in Section 5.4.2 but with a smaller evaluation budget than the budget listed in Table 5.4: The size of the initial dataset S_* is set to 10, 60 for MOEA/D-EGO(EB) and comparison algorithms, respectively. 30 extra evaluations for further optimisation are allowed. The total evaluation budget is 40, 90 for MOEA/D-EGO(EB) and comparison algorithms, respectively.

The comparison results reported in Table 5.7 and Fig. 5.4 show that MOEA/D-

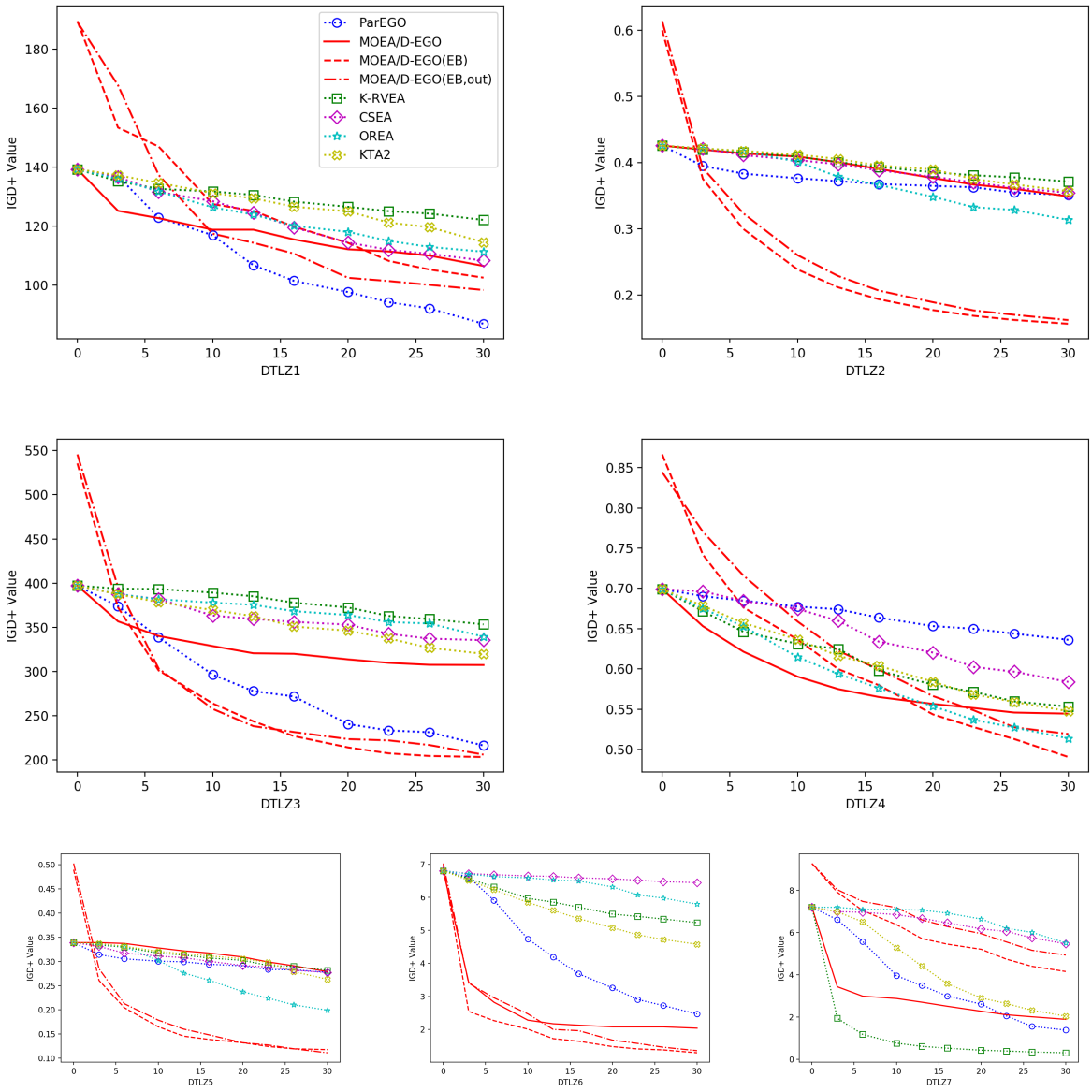


Figure 5.4: Mean Inverted Generational Distance Plus (IGD+) values of 30 runs on the optimisation of DTLZ test problems. MOEA/D-EGO(EB)s and comparison algorithms initialise their surrogates with 10, 60 samples, respectively. Extra 30 evaluations are allowed in the further optimisation. Note that 'EB out' indicates the target task is excluded from the range of related tasks during the meta-learning procedure. X-axis denotes the number of evaluations used after the surrogate initialisation. In comparison to MOEA/D-EGO, both MOEA/D-EGO(EB)s achieve smaller or competitive IGD+ values on all DTLZ test problems except for DTLZ7, while 50 evaluations are saved with the assistance from related tasks. Moreover, MOEA/D-EGO(EB)s achieve the smallest IGD+ values on DTLZ2, DTLZ3, DTLZ4, DTLZ5 and DTLZ6.

EGO(EB) has achieved competitive or smaller IGD+ values than MOEA/D-EGO on all DTLZ problems except DTLZ7. Meanwhile, $5d$ evaluations have been saved. Consistent with the results discussed in the last subsection, MOEA/D-EGO(EB) fails to achieve a competitive result compared to MOEA/D-EGO on DTLZ7 since experience is learned from small datasets collected from related tasks. Although we set a different evaluation budget for all SAEAs, the size of datasets for meta-learning $|D_m|$ has not been modified. However, it can be observed from the statistical test results (see the last row of Tables 5.5 and 5.7) that MOEA/D-EGO(EB) outperforms the comparison algorithms on 26, 29 test instances when the total evaluation budget of comparison algorithms is set to 150, 90, respectively. This answers the question we raised before: The advantage of our experience-based SAEA framework is more prominent in the extremely expensive problems where a smaller evaluation budget is allowed. The comparison between the results obtained from Tables 5.5 and 5.7 has demonstrated that our SAEA framework is preferable when solving optimisation problems within a very limited evaluation budget.

5.4.4 Influence of Task Similarity

In real-world applications, it is optimistic to assume some related tasks are very similar to the target task. A more common situation is that all related tasks have limited similarity to the target task. To investigate the relationship between task similarity and SAEA optimisation performance, we also test the performance in an 'out-of-range' situation, where the original DTLZ is excluded from the range of DTLZ variants during the MDKL meta-learning procedure. As a result, only the DTLZ variants that are quite different from the original DTLZ problem can be used for experience learning. The 'out-of-range' situation eliminates the probability that MDKL surrogates benefit greatly from the DTLZ variants that are very similar to the original DTLZ problem. Detailed definitions of the related tasks used in the 'out-of-range' situation are given in Section 5.4.2 (Generation of DTLZ variants). Apart

Table 5.8: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 runs on the optimisation of DTLZ test problems. 'Out-of-range' indicates the target task is excluded from the range of related tasks during the meta-learning procedure. Both MOEA/D-EGO(EB)s initialise their surrogates with 10, extra 50 evaluations are allowed in the further optimisation. '+', '≈', and '−' denote the result of the 'out-of-range' situation is statistically significantly superior to, almost equivalent to, and inferior to that of the 'in-range' situation in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last two rows count the statistical test results between MOEA/D-EGO(EB)s and other compared algorithms.

MOEA/D-EGO(EB)s	In-range	Out-of-range
DTLZ1	9.70e+1(1.87e+1)≈	9.11e+1(1.53e+1)
DTLZ2	1.43e-1(2.29e-2)≈	1.41e-1(1.75e-2)
DTLZ3	1.97e+2 (1.64e+1)≈	1.98e+1(1.51e+1)
DTLZ4	4.44e-1(1.35e-1)≈	4.96e-1(8.63e-2)
DTLZ5	1.13e-1(2.24e-2)≈	1.03e-1(2.39e-2)
DTLZ6	1.11e+0(5.71e-1)≈	1.17e+0(6.88e-1)
DTLZ7	2.47e+0(1.89e+0)≈	2.86e+0(1.87e+0)
+/ ≈ /−	0/7/0	-/-/-
vs MOEA/D-EGO	4/2/1	4/2/1
vs 6 Comparisons	26/9/7	27/7/8

from the related tasks used, the remaining experimental setups are the same as the setups described in Sections 5.4.2 and 5.4.3. For the sake of convenience, we denote the situation we tested in Sections 5.4.2 and 5.4.3 as 'in-range' below.

The statistical test results reported in Tables 5.8 and 5.9 show that the 'out-of-range' situation achieves competitive IGD+ values to the 'in-range' situation on all 7 test instances. This suggests that the related tasks that are very similar to the target task have a limited impact on the optimisation performance of our experience-based SAEA framework. Useful experience can be learned from the related tasks that are not very similar to the target task. Crucially, when comparing the performance of the 'out-of-range' situation and that of MOEA/D-EGO, we can still observe competitive or improved optimisation results on 6 DTLZ problems (see Tables 5.8 and 5.9, the row titled by 'vs MOEA/D-EGO', or Figs. 5.3 and 5.4). Moreover, it can be seen from the last row of Table 5.8 that the 'out-of-range' situation achieves better/competitive/worse IGD+ values than all compared SAEAs on 27/7/8 test instances. In comparison, the 'in-range' situation achieves better/competitive/worse IGD+

Table 5.9: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 runs on the optimisation of DTLZ test problems. 'Out-of-range' indicates the target task is excluded from the range of related tasks during the meta-learning procedure. Both MOEA/D-EGO(EB)s initialise their surrogates with 10, extra 30 evaluations are allowed in the further optimisation. '+', '≈', and '−' denote the result of the 'out-of-range' situation is statistically significantly superior to, almost equivalent to, and inferior to that of the 'in-range' situation in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last two rows count the statistical test results between MOEA/D-EGO(EB)s and other compared algorithms.

MOEA/D-EGO(EB)s	In-range	Out-of-range
DTLZ1	1.03e+2(2.34e+1)≈	9.84e+1(2.04e+1)
DTLZ2	1.57e-1(2.29e-2)≈	1.62e-1(1.90e-2)
DTLZ3	2.03e+2(2.42e+1)≈	2.06e+2(2.13e+1)
DTLZ4	4.91e-1(1.24e-1)≈	5.20e-1(6.92e-2)
DTLZ5	1.18e-1(2.25e-2)+	1.11e-1(2.41e-2)
DTLZ6	1.29e+0(6.44e-1)≈	1.36e+0(7.36e-1)
DTLZ7	4.16e+0(2.54e+0)≈	4.94e+0(2.31e+0)
+/ ≈ /−	0/7/0	-/-/-
vs MOEA/D-EGO	4/2/1	4/2/1
vs 6 Comparisons	29/8/5	28/9/5

values than all compared SAEAs on 26/9/7 test instances. The difference between these statistical test results is not significant. Furthermore, similar results can be observed in the last row of Table 5.9, there is only a minor difference between the optimisation results of two situations.

Consequently, the related tasks that are very similar to the target task are not essential to the optimisation performance of our experience-based SAEA framework on these problems. In the 'out-of-range' situation, our MOEA/D-EGO(EB) can still reach competitive or better optimisation results than MOEA/D-EGO while using only $1d$ samples for surrogate initialisation. The related tasks that are very similar to the target task are not necessary to the application of our SAEA framework.

Table 5.10: Mean Inverted Generational Distance Plus (IGD+) and standard deviation (in brackets) of 30 MOEA/D-EGO(EB) runs on the optimisation of DTLZ test problems. 10 samples are used for initialisation and extra 50 evaluations are allowed in the further optimisation. $|D_m|$ is the size of the dataset collected from each related task. '+', '≈', and '−' denote the result of $|D_m| = 60$ is statistically significantly superior to, almost equivalent to, and inferior to that of $|D_m| = 20$ in the Wilcoxon rank sum test (significance level is 0.05), respectively. The last row counts the total win/tie/loss results. It shows that using a large D_m for the meta-learning procedure can improve the optimisation performance of experience-based SAEA framework.

Problem	In-range		Out-of-range	
	$\|D_m\|$=20	$\|D_m\|$=60	$\|D_m\|$=20	$\|D_m\|$=60
DTLZ1	9.70e+1(1.87e+1)≈	9.77e+1(1.73e+1)	9.11e+1(1.53e+1)≈	9.93e+1(1.87e+1)
DTLZ2	1.43e-1(2.29e-2)+	1.24e-1(2.11e-2)	1.41e-1(1.75e-2)+	1.29e-1(2.36e-2)
DTLZ3	1.97e+2 (1.64e+1)≈	1.98e+2 (2.21e+1)	1.98e+1(1.51e+1)≈	1.93e+2(1.19e+1)
DTLZ4	4.44e-1(1.35e-1)≈	5.17e-1(5.68e-2)	4.96e-1(8.63e-2)≈	5.17e-1(5.38e-2)
DTLZ5	1.13e-1(2.24e-2)+	9.96e-2(2.18e-2)	1.03e-1(2.39e-2)≈	1.05e-1(2.73e-2)
DTLZ6	1.11e+0(5.71e-1)≈	1.04e+0(6.06e-1)	1.17e+0(6.88e-1)≈	1.22e+0(6.41e-1)
DTLZ7	2.47e+0(1.89e+0)+	7.49e-1(2.61e-1)	2.86e+0(1.87e+0)+	6.96e-1(2.41e-1)
+/ ≈ /−	3/4/0	-/-/-	2/5/0	-/-/-

5.4.5 Influence of the Size of Datasets Used in Meta-Learning

We also investigated the performance of our experience-based SAEA framework when different sizes of datasets $|D_m|$ are used in the meta-learning procedure. The experimental setups are the same as the setups of MOEA/D-EGO(EB) in Section 5.4.2 except for $|D_m|$.

It is evident from Table 5.10 that when each DTLZ variant provides $|D_m| = 60$ samples for the meta-learning of MDKL surrogates, the performance of both MOEA/D-EGO(EB)s are improved on 2 or 3 DTLZ problems. Thus, it would be desirable if relatively large datasets D_m can be sampled from related tasks and then used in the meta-learning procedure. Particularly, a significant improvement can be observed from the optimisation results of DTLZ7. As we discussed in Section 5.4.2, the poor performance of our experience-based optimisation on DTLZ7 is caused by the small $|D_m|$. Optimal solutions have few chances to be included in a small D_m, which makes D_m fails to provide the experience about the discontinuity of optimal regions. In comparison, the experience of 'optimal regions' can be learned from large datasets D_m and thus the optimisation results are improved significantly.

In conclusion, for our experience-based SAEA framework, a large $|D_m|$ for the meta-learning procedure indicates more valuable experience can be learned from related tasks, which further improves the performance of experience-based optimisation. Therefore, when applying our experience-based SAEA framework to real-world optimisation problems, it is preferable to collect more data from related tasks for experience learning.

5.4.6 Performance on Expensive Constrained Optimisation (Engine Calibration Problem)

The experiments on multi-objective benchmark test problems have investigated the performance of our SAEA framework **in depth**. In this subsection, we finally study a real-world gasoline motor engine calibration problem, which is an expensive constrained single-objective optimisation problem. This experiment covers the optimisation scenarios of single-objective optimisation, constrained optimisation, and real-world applications. Hence, it serves as an example to demonstrate the **generality** and **broad applicability** of our experience-based SAEA framework.

The calibration problem has 6 adjustable engine parameters including throttle angle, waste gate orifice, ignition timing, valve timings, state of injection, and air-fuel-ratio. The calibration aims at minimising the BSFC and satisfying 4 constraints in terms of temperature, pressure, CA50, and load simultaneously [123].

Comparison algorithms:

Since the comparison algorithms in the DTLZ optimisation experiments are not designed for handling constrained single-objective optimisation, our comparison is conducted with two state-of-the-art constrained optimisation algorithms used in industry [123]: a variant of EGO designed to handle constrained optimisation problems (denoted by constrained_EGO), and a GA customised for this calibration problem (denoted by adaptiveGA). The settings of

comparison algorithms are the same as suggested in [123]. In this experiment, we apply our SAEA framework to constrained_EGO and investigate its optimisation performance. The GP surrogates in constrained_EGO are replaced by our MDKL surrogates to conduct the comparison, and the resulting algorithm is denoted as constrained_EGO(EB).

Experimental setups:

The setups of related tasks (N, D_i) are the same as described in Section 5.4.1. In the meta-learning procedure, both support set and query set contain 6 data points, thus $|D_m| = 12$. The total evaluation budget for all algorithms is set to 60. For adaptiveGA, all evaluations are used in the optimisation process as it is not a SAEA. For constrained_EGO, 40 samples are used to initialise the surrogates and 20 extra evaluations are used in the optimisation process. For constrained_EGO(EB), only 6 samples are used to initialise MDKL surrogates, and the remaining evaluations are used for further optimisation.

Optimisation results and analysis:

The mean normalised BSFC results and the mean number of feasible solutions found over the number of evaluations used are plotted in Figs. 5.5a and 5.5c, respectively. The statistical results of three comparison algorithms are illustrated in Figs. 5.5b and 5.5d. From Fig. 5.5a, it can be observed that the minimal BSFC obtained by constrained_EGO(EB) decreases drastically in the first few evaluations, implying the experience learned from related tasks is working. In comparison, the minimal BSFC obtained by adaptiveGA and constrained_EGO drops in a relatively slow rate, even though constrained_EGO has used 34 more samples to initialise its surrogates. The star marker denotes the point at which constrained_EGO(EB) has evaluated 20 samples after surrogate initialisation. It is worth noting that when 20 samples have been evaluated in the optimisation, constrained_EGO(EB) achieves a smaller BSFC value than constrained_EGO. In addition, after the star marker, the decrease of BSFC becomes slow as constrained_EGO(EB) has reached the optimal region that we found so

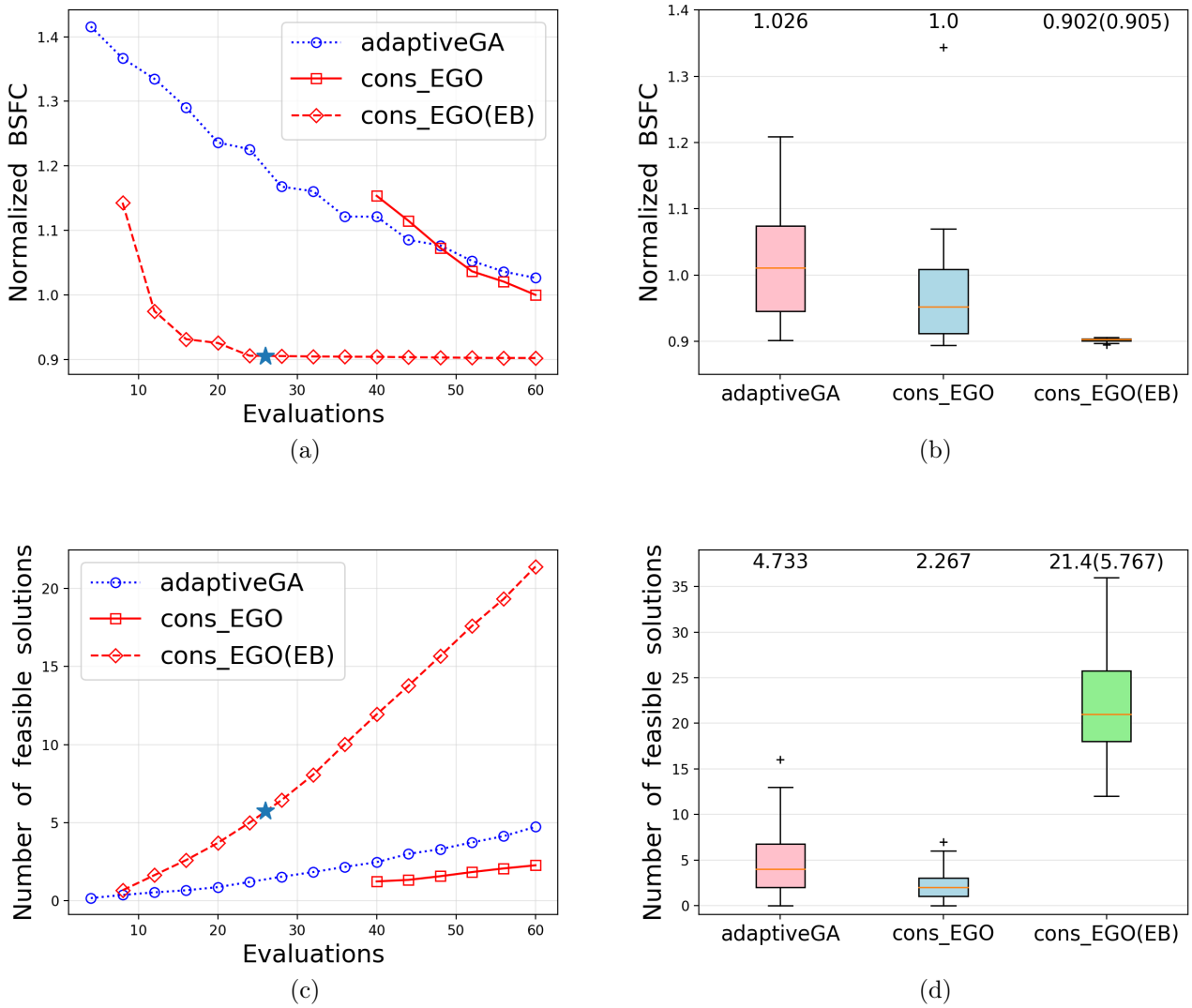


Figure 5.5: Results of 30 runs on the engine calibration problem, all BSFC values are normalised. The evaluation budget is set to 60, including 40, 6 samples used to initialise surrogates for constrained_EGO and its experience-based variant, respectively. Figs. (a) and (c) show how BSFC and the number of feasible solutions vary with the number of evaluations, respectively. The star markers highlight the results achieved when 20 evaluations are used in the optimisation process. It can be seen that constrained_EGO(EB) can achieve a smaller BSFC value and find more feasible solutions than the compared algorithms, indicating the experience learned from related tasks is effective in objective function and constraint functions. Figs. (b) and (d) illustrate the statistical results of BSFC and the number of feasible solutions when the evaluation budget has run out. Mean values are shown on the top, the results in brackets are achieved at the star markers of Figs. (a) and (c).

far. Therefore, the further improvement in the normalised BSFC value is not significant and thus hard to be observed. The advantages of our experience-based SAEA framework can also be observed in constraint handling. In Figs. 5.5c and 5.5d, constrained_EGO(EB) finds more feasible solutions than the two comparison algorithms. These results indicate that our SAEA framework improves the performance of constrained_EGO on both objective function and constraint functions. Meanwhile, only $1d$ evaluations are used to initialise surrogates.

Discussion on runtime:

It should be noted that real engine performance evaluations on engine facilities are very costly in terms of both time and financial budget [116]. Since a single real engine performance evaluation can cost several hours [59, 116], the time cost of the meta-learning procedure is negligible as it takes only a few minutes. Savings from reduced real engine performance evaluations on engine facilities and the reduced development cycle due to our SAEA framework could amount to millions of dollars [116]. Therefore, our SAEA framework is an effective and efficient method to solve this real-world calibration problem.

5.4.7 Discussions

Our computational studies have demonstrated the following: First, we provide empirical evidence to show the effectiveness of learning experience: The meta-learning of neural network parameters and base kernel parameters are essential to the modelling accuracy of a MDKL model. As a result, our MDKL model outperforms the compared meta-learning modelling and non-meta-learning modelling methods on both the engine fuel consumption regression task and the sinusoid function regression task.

Second, we demonstrate the main contribution of this work: In most situations, our proposed experience-based SAEA framework can assist regression-based SAEAs to reach competitive or even better optimisation results, while the cost of surrogate initialisation is

only $1d$ samples. Due to the effectiveness of saving evaluations, our SAEA framework is preferable to other SAEAs when solving problems within a very limited evaluation budget. Moreover, some empirical guidelines are concluded to help the application of our SAEA framework. For the influence of task similarity, we find that related tasks that are very similar to the target task are not necessary to the application of our approach. The influence of these similar tasks on the optimisation performance is limited. Our experience-based SAEA framework can achieve competitive results without datasets from very similar related tasks. In addition, for the related tasks used for meta-learning, we have demonstrated that more useful experience can be learned if more data points are sampled from related tasks.

Third, the effectiveness of our SAEA framework is validated on a real-world engine calibration problem. Competitive or better results are achieved on the objective and constraint functions, while $1d$ samples are used to initialise surrogates. Therefore, our experience-based SAEA framework can also be applied to optimisation scenarios such as single-objective optimisation and constrained optimisation.

5.5 Chapter Summary

Experienced human engineers are good at tackling new and unseen optimisation problems in comparison to novices. There has been an ongoing effort in evolutionary optimisation trying to capture and then use such experience [57, 96, 95]. In this chapter, we present an experience-based SAEA framework to solve expensive optimisation problems. To learn experience from related tasks, a novel meta-learning modelling method, namely MDKL, has been developed. Our MDKL model learns the domain-specific features of a set of related tasks from plenty of small datasets. The learned experience is integrated with very limited examples collected from the target optimisation problem, which improves the modelling efficiency and approximation accuracy. The effectiveness of learning experience has been demonstrated by comparing our MDKL modelling method with other meta-learning mod-

elling and non-meta-learning modelling methods. Our experience-based SAEA framework uses MDKL models as surrogates and an MSE-based update criterion is proposed to manage MDKL surrogates during the evolutionary optimisation. Our SAEA framework is applicable to any regression-based SAEAs by replacing their original surrogates with our MDKL surrogates. Our computational studies have demonstrated the effectiveness of our SAEA framework on multi-objective optimisation and constrained single-objective optimisation (a real-world engine application). On most test problems, better or competitive optimisation results are achieved when only $1d$ samples are used to initialise our surrogates. It has been demonstrated that the advantage of our SAEA framework becomes more prominent on the extremely expensive problems where very few evaluations are allowed.

The contributions of this chapter can be summarised as:

- A novel meta-learning method (Meta Deep Kernel Learning, MDKL) is developed to learn experience from related expensive tasks. Based on the learned experience, a regression-based surrogate is generated and then adapted to approximate the fitness landscape of the target task. The surrogate is derived from the deep kernel learning framework in which a Gaussian process employs a deep kernel to work as its covariance function. Different from existing deep kernel learning models that are trained through meta-learning, our method learns only one common neural network for all related tasks and adapts task-specific parameters for the base kernel of a Gaussian process. Such a framework simplifies the architecture of experience learning as the model complexity will not grow with the number of past tasks, yet it is still able to adapt itself to a new task explicitly. This is an answer to the first question in Section 1.3.3.

- We propose an experience-based SAEA framework to combine regression-based SAEAs with our experience learning method. In the framework, our meta-learning model is employed as surrogates to learn experience and an update strategy is designed to adapt surrogates constantly. Such a framework is compatible with existing regression-based

SAEAs, making these SAEAs can be assisted by the experience from related tasks. Note that our SAEA framework is a general framework as it is not designed for any specific scenario of optimisation problems, such as single-objective optimisation, multi-objective optimisation, constrained optimisation, large-scale optimisation, or combinatorial optimisation. However, due to the page limitation, this thesis focuses on only the scenarios of multi-objective optimisation and constrained optimisation, which have not been investigated before This is an answer to the second question in Section 1.3.3.

Chapter 6

Conclusions and Future Work

This chapter concludes the contributions of this thesis and discusses some potential directions for future work related to the work presented in this thesis.

6.1 Conclusions

This thesis is dedicated to develop novel and efficient SAEAs to address expensive optimisation problems, especially for EMOPs and ECOPs. The contents presented in Chapters 3-5 have answered the research questions we raised in Section 1.3.

In Chapter 3, two ordinal-regression-based SAEAs, OREA and AOREA, are proposed to address EMOPs. An ordinal-regression-based surrogate is developed to approximate the dominance-based ordinal landscape for two SAEAs. This ordinal-regression-based surrogate is computationally efficient since the landscape of multi-dimensional objective space is approximated using only one surrogate. Therefore, our research question "How to construct an efficient surrogate to approximate the ordinal landscape of multi-objective optimisation problems?" is answered. Our computational studies on DTLZ benchmark test problems show that OREA outperforms state-of-the-art comparison SAEAs in terms of IGD+ values when a limited fitness evaluation budget is available. Such an observation indicates that

the hybrid of individual-based and generation-based evolution control (we used in OREA) is an effective surrogate management strategy to balance convergence and diversity for the ordinal-regression-based surrogate. In addition, AOREA achieves smaller IGD+ values on EMOPs than OREA and other comparison SAEAs, implying that the adaptive surrogate management strategy used in AOREA is preferable to the hybrid one. In other words, it is desirable to adapt the balance between convergence and diversity dynamically as the evolutionary optimisation state changes. The observations discussed above have answered our research question "How to balance convergence and diversity when using our ordinal surrogate in SAEAs to solve EMOPs".

Chapter 4 takes constraints into consideration and focuses on addressing a real-world ECOP, gasoline motor engine calibration. In many real-world applications, the feasibility of solutions are mainly affected by a subset of all decision variables. Inspired by this phenomenon, a surrogate-assisted bilevel evolutionary algorithm, namely SAB-DE, is proposed to use a bilevel architecture to solve real-world ECOPs. In the bilevel architecture, decision variables that have significant impacts on solution feasibility would be divided into lower-level variables and then optimised to handle constraints, while remaining decision variables are classified as upper-level variables and would be adjusted to optimise objective(s). To improve the efficiency of handling constraints, a PCA-based sensitivity analysis method is developed to quantify the impact of decision variables on solution feasibility. Our experimental studies show that our sensitivity analysis method are capable of identifying the decision variables which have significant impacts on solution feasibility, resulting in a reasonable division of upper-level and lower-level variables. In addition, in lower-level optimisation, our ordinal-regression-based surrogate is adapted to approximate the ordinal landscape of constraints. The reported comparison experiments have demonstrated that using an ordinal-regression-based surrogate can reach a higher prediction accuracy on solution feasibility than using multiple constraint approximation surrogates or using a surrogate to approximate the total violation of constraints. Moreover, the computational efficiency of using an ordinal-

regression-based surrogate to handle multiple constraints has been demonstrated. Finally, it can be observed from the engine calibration experiments that our SAB-DE and $\epsilon-$selection environmental selection strategy are effective. SAB-DE achieves better BSFC results and also find more feasible solutions than compared expensive constrained optimisation algorithms. The contributions concluded above have answered the research questions listed in Section 1.3.2.

To endow SAEAs with the capability of learning experience and make SAEAs more intelligent, in Chapter 5, an experience-based optimisation framework is proposed to address very expensive optimisation problems. In this experience-based optimisation framework, a meta-learning method, called MDKL, is developed to learn experience from related expensive optimisation problems. Our studies show that the MDKL model is effective in learning experience and all MDKL components have contributed to the performance of learning experience. An update strategy is designed to use the MDKL model in the experience-based optimisation framework. The performance of experience-based expensive optimisation is tested on two representative expensive optimisation scenarios: EMOPs and ECOPs, which are the research topics this thesis studied in previous chapters. Our experimental studies have demonstrated that the idea of experience-based optimisation works for expensive optimisation, especially for very expensive optimisation where only a few fitness evaluations are allowed. In comparison to state-of-the-art SAEAs, experience-based SAEAs have achieved better or competitive optimisation results with a cost of fewer fitness evaluations, showing its effectiveness and efficiency on addressing (very) expensive optimisation problems. In addition, some empirical guidances are concluded to help the application of experience-based expensive optimisation in real-world applications. Our studies show that very similar and related optimisation problems are not necessary for learning experience. Moreover, increasing the size of datasets collected from related optimisation problems is beneficial to the learning of useful experience.

6.2 Future Work

There are four possible research directions for future work: compatibility, generalisation, development, and completeness.

From the perspective of compatibility, the compatibility of our proposed ordinal-regression-based surrogate and other SAEAs can be studied. Also, the compatibility of the adaptive management strategy and other SAEAs can be studied. Furthermore, the surrogates employed in SAB-DE are Kriging models. Considering diverse surrogate modelling methods have been developed to approximate the objective and constraint functions [100], the performance of combinations of SAB-DE framework and different surrogates could be studied to discover the most appropriate and compatible combination.

To investigate the generalisation of our methodologies, it would be interesting to examine the performance of our methodologies in more expensive optimisation scenarios. For example, the optimisation performance of OREA and AOREA for expensive many-objective optimisation problems can be investigated. Besides, the engine calibration problem solved in Chapter 4 has only one objective, and the performance of SAB-DE on expensive multi-objective constrained optimisation problems needs further investigation. Additionally, while the constraints discussed in the engine calibration problem are all inequality constraints. Equality constraints need to be considered in the future. Furthermore, different types of real-world engines [116, 98] should be considered in our study to evaluate the effectiveness of the SAB-DE framework. And it would be interesting to investigate whether the bilevel architecture could serve as a generic constraint handling method for constrained optimisation in general.

From the view of algorithm development, developing new methodologies to further improve the optimisation performance of the methods proposed in this thesis could be a direction of future work. For our adaptive surrogate management strategy, different criteria can be developed to update the state of global search. For the SAEA without reference vec-

tors, more diversity maintenance methods can be designed and introduced to the adaptive management strategy. Also, as discussed in the last paragraph of Section 3.4, the distribution of non-dominated solutions is not perfect and there are some spaces to improve it. Additionally, some of the techniques developed for SAEAs for single-objective optimisation [100] could be adapted to the multi-objective case.

Finally, considering the completeness of research, future work could be conducted to address the open research questions that have not solved in this thesis. For example, the precise definition of experience is still unclear. Different work so far seems to have captured different aspects of experience. A more precise and comprehensive definition of optimisation experience is needed. It is also a challenging task to represent such experience formally. As regard to the SAEA work in Chapter 5, there are two specific future research directions. First, we do not have a mathematical definition of related tasks. Although our computational studies have demonstrated that the experience learned from a set of related tasks that differ from the target task is beneficial to the optimisation, we cannot guarantee the optimisation performance if we further decrease the similarity between the related tasks and the target task. It is very interesting to study similarity measures between tasks in the context of experience-based SAEA framework. In fact, this is also a general research topic that is relevant to other experience-based optimisation methods, such as transfer optimisation [78, 79]. Second, the proposed framework is currently for regression-based SAEAs only. As we can see from the DTLZ optimisation experiments, classification-based SAEAs and ordinal-regression-based SAEAs could be more effective sometimes. These SAEAs learn surrogates from user-assigned class labels or ordinal relationships, instead of original fitness values. It is an interesting future work to do meta-learning from user-assigned values.

Bibliography

[1] Haldun Aytuğ and Serpil Sayın. Using support vector machines to learn the efficient set in multiple objective discrete optimization. *European Journal of Operational Research*, 193(2):510–519, 2009.

[2] Kavitesh Kumar Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. *IEEE Transactions on Evolutionary Computation*, 24(1):69–83, 2019.

[3] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[4] Jürgen Branke and Christian Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9:13–20, 2005.

[5] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.

[6] Ji Cheng, Ping Jiang, Qi Zhou, Jiexiang Hu, and Leshi Shu. A parallel constrained lower confidence bounding approach for computationally expensive constrained optimization problems. *Applied Soft Computing*, 106:107276, 2021.

[7] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[8] Tinkle Chugh, Yaochu Jin, Kaisa Miettinen, Jussi Hakanen, and Karthik Sindhya. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):129–142, 2016.

[9] Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, 2019.

[10] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the 1st European Conference on Artificial Life (ECAL'91)*, volume 142, pages 134–142, 1991.

[11] Ivo Couckuyt, Dirk Deschrijver, and Tom Dhaene. Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *Journal of Global Optimization*, 60(3):575–594, 2014.

[12] Rituparna Datta and Rommel G Regis. A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Systems with Applications*, 57:270–284, 2016.

[13] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338, 2000.

[14] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[15] Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[16] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T.A.M.T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[17] Kalyanmoy Deb and Ankur Sinha. Solving bilevel multi-objective optimization problems using evolutionary algorithms. In *Proceedings of the 2009 International Conference on Evolutionary Multi-Criterion Optimization (EMO'09)*, pages 110–124, 2009.

[18] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pages 105–145. Springer, London, U.K., 2005.

[19] Jinliang Ding, Cuie Yang, Yaochu Jin, and Tianyou Chai. Generalized multitasking for evolutionary optimization of expensive problems. *IEEE Transactions on Evolutionary Computation*, 23(1):44–58, 2017.

[20] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN'95)*, pages 1942–1948, 1995.

[21] M. A. El-Beltagy and A. J. Keane. Optimisation for multilevel problems: A comparison of various algorithms. In *Adaptive Computing in Design and Manufacture*, pages 111–120. Springer, London, U.K., 1998.

[22] David Eriksson and Matthias Poloczek. Scalable constrained Bayesian optimization. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS'21)*, pages 730–738, 2021.

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, pages 1126–1135, 2017.

[24] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, NY, 1966.

[25] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[26] Zhendong Guo, Haitao Liu, Yew-Soon Ong, Xinghua Qu, Yuzhe Zhang, and Jianmin Zheng. Generative multiform Bayesian optimization. *IEEE Transactions on Cybernetics*, 53(7):4347–4360, 2022.

[27] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Insights on transfer optimization: Because experience is the best teacher. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):51–64, 2017.

[28] Ahsanul Habib, Hemant Kumar Singh, Tinkle Chugh, Tapabrata Ray, and Kaisa Miettinen. A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive multi/many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23(6):1000–1014, 2019.

[29] Guanghong Han and Xi Chen. A bi-level differential evolutionary algorithm for constrained optimization. In *Proceedings of the 21st IEEE Congress on Evolutionary Computation (CEC'19)*, pages 1628–1633, 2019.

[30] Stephanus Daniel Handoko, Chee Keong Kwoh, and Yew-Soon Ong. Feasibility structure modeling: An effective chaperone for constrained memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 14(5):740–758, 2010.

[31] Hao Hao, Aimin Zhou, Hong Qian, and Hu Zhang. Expensive multiobjective optimization by relation learning and prediction. *IEEE Transactions on Evolutionary Computation*, 26(5):1157–1170, 2022.

[32] James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online Bayesian regression. In *Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics (WAFR'18)*, pages 318–337, 2018.

[33] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314, 1962.

[34] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992.

[35] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[36] Afzal Husain and Kwang-Yong Kim. Enhanced multi-objective optimization of a microchannel heat sink through evolutionary algorithm coupled with multiple surrogate models. *Applied Thermal Engineering*, 30(13):1683–1691, 2010.

[37] Anke K. Hutzschenreuter, Peter A.N. Bosman, and Han La Poutré. Evolutionary multiobjective optimization for dynamic hospital resource management. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization (EMO'09)*, pages 320–334, 2009.

[38] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In *Proceedings of the 8th International Conference on Evolutionary Multi-criterion Optimization (EMO'15)*, pages 110–125, 2015.

[39] M. Janga Reddy and D. Nagesh Kumar. Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: A state-of-the-art review. *H2Open Journal*, 3(1):135–188, 2021.

[40] Min Jiang, Zhenzhong Wang, Shihui Guo, Xing Gao, and Kay Chen Tan. Individual-based transfer learning for dynamic multiobjective optimization. *IEEE Transactions on Cybernetics*, 51(10):4968–4981, 2020.

[41] Min Jiang, Zhenzhong Wang, Liming Qiu, Shihui Guo, Xing Gao, and Kay Chen Tan. A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning. *IEEE Transactions on Cybernetics*, 51(7):3417–3428, 2020.

[42] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[43] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[44] Yaochu Jin, Sanghoun Oh, and Moongu Jeon. Incremental approximation of nonlinear constraint functions for evolutionary constrained optimization. In *Proceedings of the 12th IEEE Congress on Evolutionary Computation (CEC'10)*, pages 1–8, 2010.

[45] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO'00)*, pages 786–793, 2000.

[46] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[47] Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[48] Teuvo Kohonen. An introduction to neural computing. *Neural Networks*, 1(1):3–16, 1988.

[49] John R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[50] Genghui Li and Qingfu Zhang. Multiple penalties and multiple local surrogates for expensive constrained optimization. *IEEE Transactions on Evolutionary Computation*, 25(4):769–778, 2021.

[51] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Sam Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716, 2014.

[52] JJ. Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8):8–31, 2006.

[53] Rung-Tzuo Liaw and Chuan-Kang Ting. Evolutionary manytasking optimization based on symbiosis in biocoenosis. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, pages 4295–4303, 2019.

[54] Dudy Lim, Yaochu Jin, Yew-Soon Ong, and Bernhard Sendhoff. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355, 2009.

[55] Lin Lin and Mitsuo Gen. Hybrid evolutionary optimisation with learning for production scheduling: State-of-the-art survey on algorithms and applications. *International Journal of Production Research*, 56(1-2):193–223, 2018.

[56] Bo Liu, Qingfu Zhang, and Georges GE Gielen. A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(2):180–192, 2013.

[57] Shengcai Liu, Ke Tang, and Xin Yao. Experience-based optimization: A coevolutionary approach. *arXiv preprint arXiv:1703.09865*, 2017.

[58] Yuanhao Liu, Zan Yang, Danyang Xu, Haobo Qiu, and Liang Gao. A surrogate-assisted differential evolution for expensive constrained optimization problems involving mixed-integer variables. *Information Sciences*, 622:282–302, 2023.

[59] He Ma. *Control Oriented Engine Modeling and Engine Multi-objective Optimal Feedback Control*. PhD thesis, School of Mechanical Engineering, University of Birmingham, Birmingham, U.K., 2013.

[60] Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[61] Efrén Mezura-Montes and Carlos A Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17, 2005.

[62] Mariana-Edith Miranda-Varela and Efrén Mezura-Montes. Surrogate-assisted differential evolution with an adaptive evolution control based on feasibility to solve constrained optimization problems. In *Proceedings of the 5th International Conference on Soft Computing for Problem Solving (SocProS'16)*, pages 809–822, 2016.

[63] Mariana-Edith Miranda-Varela and Efrén Mezura-Montes. Constraint-handling techniques in surrogate-assisted evolutionary optimization. an empirical study. *Applied Soft Computing*, 73:215–229, 2018.

[64] Kishalay Mitra and Sushanta Majumder. Successive approximate model based multi-objective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm. *Chemical Engineering Science*, 66(15):3471–3481, 2011.

[65] Pawan KS Nain and Kalyanmoy Deb. Computationally effective search and optimization procedure using coarse to fine approximations. In *Proceedings of the 5th IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 2081–2088, 2003.

[66] P. B. Nair, A. J. Keane, and R. P. Shimpi. Combining approximation concepts with genetic algorithm-based structural optimization procedures. In *Proceedings of the 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit*, page 1912, 1998.

[67] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 807–814, 2010.

[68] Linqiang Pan, Cheng He, Ye Tian, Handing Wang, Xingyi Zhang, and Yaochu Jin. A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):74–88, 2018.

[69] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In *Advance in Neural Information Processing Systems 33 (NeurIPS'20)*, 2020.

[70] Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

[71] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN X)*, pages 784–794, 2008.

[72] Sultan Noman Qasem, Siti Mariyam Shamsuddin, Siti Zaiton Mohd Hashim, Maslina Darus, and Eiman Al-Shammari. Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems. *Information Sciences*, 239:165–190, 2013.

[73] Jiachang Qian, Yuansheng Cheng, Jinlan Zhang, Jun Liu, and Dawei Zhan. A parallel constrained efficient global optimization algorithm for expensive constrained optimization problems. *Engineering Optimization*, 53(2):300–320, 2021.

[74] Kamrul Hasan Rahi, Hemant Kumar Singh, and Tapabrata Ray. A steady-state algorithm for solving expensive multi-objective optimization problems with non-parallelizable evaluations. *IEEE Transactions on Evolutionary Computation*, 27(5):1544–1558, 2022.

[75] Rommel G. Regis. Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. *IEEE Transactions on Evolutionary Computation*, 18(3):326–347, 2013.

[76] Rommel G. Regis. Multi-objective constrained black-box optimization using radial basis function surrogates. *Journal of Computational Science*, 16:140–155, 2016.

[77] Rommel G. Regis. A survey of surrogate approaches for expensive constrained black-box optimization. In *Proceedings of the 6th World Congress on Global Optimization (WCGO'19)*, pages 37–47, 2019.

[78] Gan Ruan, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. When and how to transfer knowledge in dynamic multi-objective optimization. In *Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI'19)*, pages 2034–2041, 2019.

[79] Gan Ruan, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao. Computational study on effectiveness of knowledge transfer in dynamic multi-objective optimization. In *Proceedings of the 22nd IEEE Congress on Evolutionary Computation (CEC'20)*, pages 1–8, 2020.

[80] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[81] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000.

[82] Thomas Philip Runarsson. Constrained evolutionary optimization by approximate ranking and surrogate models. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 401–410, 2004.

[83] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[84] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, NY, 1995.

[85] Chun-Wei Seah, Yew-Soon Ong, Ivor W. Tsang, and Siwei Jiang. Pareto rank learning in multi-objective evolutionary algorithms. In *Proceedings of the 14th IEEE Congress on Evolutionary Computation (CEC'12)*, pages 1–8, 2012.

[86] Palwasha W. Shaikh, Mohammed El-Abd, Mounib Khanafer, and Kaizhou Gao. A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):48–63, 2020.

[87] Liang Shi and Khaled Rasheed. Asaga: An adaptive surrogate-assisted genetic algorithm. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*, pages 1049–1056, 2008.

[88] Hemant Kumar Singh, Tapabrata Ray, and Warren Smith. Surrogate assisted simulated annealing (SASA) for constrained multi-objective optimization. In *Proceedings of the 12th IEEE Congress on Evolutionary Computation (CEC'10)*, pages 1–8, 2010.

[89] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[90] Zhenshou Song, Handing Wang, Cheng He, and Yaochu Jin. A Kriging-assisted two-archive evolutionary algorithm for expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 25(6):1013–1027, 2021.

[91] Michael L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, New York, NY, 1999.

[92] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[93] Yuanping Su, Lihong Xu, and Erik D. Goodman. Hybrid surrogate-based constrained optimization with a new constraint-handling method. *IEEE Transactions on Cybernetics*, 52(6):5394–5407, 2020.

[94] Tetsuyuki Takahama, Setsuko Sakai, and Noriyuki Iwane. Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In *Proceedings of the 18th Australian Joint Conference on Advances in Artificial Intelligence (AUS-AI'05)*, pages 389–400, 2005.

[95] Kay Chen Tan, Liang Feng, and Min Jiang. Evolutionary transfer optimization-a new frontier in evolutionary computation research. *IEEE Computational Intelligence Magazine*, 16(1):22–33, 2021.

[96] Ke Tang, Shengcai Liu, Peng Yang, and Xin Yao. Few-shots parallel algorithm portfolio construction via co-evolution. *IEEE Transactions on Evolutionary Computation*, 25(3):595–607, 2021.

[97] Mohammad H. Tayarani-N, Adam Prugel Bennett, Hongming Xu, and Xin Yao. Improving the performance of evolutionary engine calibration algorithms with principal component analysis. In *Proceedings of the 18th IEEE Congress on Evolutionary Computation (CEC'16)*, pages 5128–5137, 2016.

[98] Mohammad-H Tayarani-N, Xin Yao, and Hongming Xu. Meta-heuristic algorithms in car engine design: A literature survey. *IEEE Transactions on Evolutionary Computation*, 19(5):609–629, 2014.

[99] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017.

[100] Hao Tong, Changwu Huang, Leandro L. Minku, and Xin Yao. Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. *Information Sciences*, 562:414–437, 2021.

[101] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

[102] Michael Volpp, Lukas P. Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *Proceedings of the 8th International Conference on Learning Representations (ICLR'20)*, 2020.

[103] Handing Wang, Licheng Jiao, and Xin Yao. Two arch2: An improved two-archive algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):524–541, 2015.

[104] Handing Wang and Yaochu Jin. A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE Transactions on Cybernetics*, 50(2):536–549, 2018.

[105] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3):1–34, 2020.

[106] Yong Wang, Da-Qing Yin, Shengxiang Yang, and Guangyong Sun. Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints. *IEEE Transactions on Cybernetics*, 49(5):1642–1656, 2018.

[107] Feng-Feng Wei, Wei-Neng Chen, Qing Li, Sang-Woon Jeon, and Jun Zhang. Distributed and expensive evolutionary constrained optimization with on-demand evaluation. *IEEE Transactions on Evolutionary Computation*, 27(3), 2022.

[108] Tingyang Wei, Shibin Wang, Jinghui Zhong, Dong Liu, and Jun Zhang. A review on evolutionary multi-task optimization: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 26(5):941–960, 2021.

[109] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT press, Cambridge, MA, 2006.

[110] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS'16)*, pages 370–378, 2016.

[111] Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*, 2021.

[112] Guohua Wu, Witold Pedrycz, Ponnuthurai N Suganthan, and Rammohan Mallipeddi. A variable reduction strategy for evolutionary algorithms handling equality constraints. *Applied Soft Computing*, 37:774–786, 2015.

[113] Xiaoming Xue, Kai Zhang, Kay Chen Tan, Liang Feng, Jian Wang, Guodong Chen, Xinggang Zhao, Liming Zhang, and Jun Yao. Affine transformation-enhanced multifactorial optimization for heterogeneous problems. *IEEE Transactions on Cybernetics*, 52(7):6217–6231, 2020.

[114] Zan Yang, Haobo Qiu, Liang Gao, Xiwen Cai, Chen Jiang, and Liming Chen. Surrogate-assisted classification-collaboration differential evolution for expensive constrained optimization problems. *Information Sciences*, 508:50–63, 2020.

[115] Xunzhao Yu, Xin Yao, Yan Wang, Ling Zhu, and Dimitar Filev. Domination-based ordinal regression for expensive multi-objective optimization. In *Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI'19)*, pages 2058–2065, 2019.

[116] Xunzhao Yu, Ling Zhu, Yan Wang, Dimitar Filev, and Xin Yao. Internal combustion engine calibration using optimization algorithms. *Applied Energy*, 305:117894, 2022.

[117] Yuan Yuan and Wolfgang Banzhaf. Expensive multi-objective evolutionary optimization assisted by dominance prediction. *IEEE Transactions on Evolutionary Computation*, 26(1):159–173, 2022.

[118] Jinyuan Zhang, Aimin Zhou, and Guixu Zhang. A classification and Pareto domination based multiobjective evolutionary algorithm. In *Proceedings of the 17th IEEE Congress on Evolutionary Computation (CEC'15)*, pages 2883–2890, 2015.

[119] Liangjie Zhang, Yuling Xie, Jianjun Chen, Liang Feng, Chao Chen, and Kai Liu. A study on multiform multi-objective evolutionary optimization. *Memetic Computing*, 13(3):307–318, 2021.

[120] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by MOEA/D with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010.

[121] Xinggang Zhao, Kai Zhang, Guodong Chen, Xiaoming Xue, Chuanjin Yao, Jian Wang, Yongfei Yang, Hui Zhao, and Jun Yao. Surrogate-assisted differential evolution for production optimization with nonlinear state constraints. *Journal of Petroleum Science and Engineering*, 194:107441, 2020.

[122] Zongzhao Zhou, Yew Soon Ong, Prasanth B. Nair, Andy J. Keane, and Kai Yew Lum. Combining global and local surrogate models to accelerate evolutionary optimization.

IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(1):66–76, 2006.

[123] Ling Zhu, Yan Wang, Anuj Pal, and Guoming Zhu. Engine calibration using global optimization methods with customization. Technical Report 2020-01-0270, SAE Technical Paper, 2020.

[124] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[125] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, pages 292–301, 1998.