Evolutionary Optimization of Expensive Multi-objective Problems with Co-sub-Pareto Front Gaussian Process Surrogates

Jianping Luo, Abhishek Gupta, Yew-Soon Ong, and Zhenkun Wang

***Abstract*—This paper proposes a Gaussian process-based *co-sub-Pareto front* (co-subPF) surrogate augmentation strategy for evolutionary optimization of computationally expensive multi-objective problems. In the proposed algorithm, a multi-objective problem is decomposed into a number of subproblems, the solution of each of which is used to approximate a portion or sector of the Pareto front (i.e., a subPF). Thereafter, a multi-task Gaussian process model is incorporated to exploit the correlations across the subproblems via joint surrogate model learning. A novel criterion for the utility function is defined on the surrogate landscape to determine the next candidate solution for evaluation using the actual expensive objectives. In addition, a new management strategy for the evaluated solutions is presented for model building. The novel feature of our approach is that it infers multiple subproblems jointly by exploiting the possible dependencies between them, such that knowledge can be transferred across subPFs approximated by the subproblems. Experimental studies under several scenarios indicate that the proposed algorithm outperforms state-of-the-art multi-objective evolutionary algorithms for expensive problems. The parameter sensitivity and effectiveness of the proposed algorithm are analyzed in detail.**

***Index Terms* — Multi-objective evolutionary algorithm, expensive optimization, multi-task Gaussian process**

I. Introduction

Optimizing several conflicting objectives simultaneously is often encountered in the field of science, engineering, business decision-making, etc. This kind of problem is normally regarded as a *multi-objective optimization problem* (MOP). In various applications, the evaluation of multiple objective functions could be extremely time consuming, and such optimization problems are generally referred to as expensive MOPs [1]. For example, in a reservoir simulation-based optimization problem, the computational cost required for merely one objective function evaluation can even be as much as several days [1].

In the last few decades, *evolutionary algorithms* (EAs) have surfaced as one of the most prominent approaches for addressing MOPs, and have achieved great success due to their simplicity and effectiveness[2-6]. However, due to the typically large number of function evaluations required by EAs, the reduction of computational burden has emerged as an important and non-trivial issue in the application of EAs to expensive MOPs. Using surrogate models to address this challenge is one of the most popular approaches. To put it simply, a *surrogate-assisted evolutionary algorithm* (SAEA) uses computationally cheap approximations of the actual objective functions to quickly progress the search. These methods have been found helpful in improving the efficiency of vanilla EAs by reducing the number of exact function evaluations in the search of high quality solutions [7], including optimization in noisy environments [8], or solving multi-modal problems with rugged fitness landscapes [9]. Some SAEAs have even been proposed to handle expensive constrained optimization problems [11-16] which are of growing practical interest. What is more, surrogate-assistance can be useful in dealing with a variety of other kinds of problems, such as in interactive evolutionary computation [17], dynamic optimization [18-19], robust optimization [20], etc. It is worth noting that, recently, some SAEAs have been applied to large-scale expensive optimization problems as well [21-23].

Gaussian Process (GP), neural networks, support vector regression, and polynomial approximation are popularly used surrogate techniques in SAEAs. They are also combined with other nature-inspired optimization techniques, such as swarm intelligence approaches, to improve problem-solving [24-34]. Among existing approaches, GP has perhaps most commonly been employed as a surrogate model in evolutionary optimization since it provides an estimate of the fitness (predicted mean of the objective function value) together with an estimate of the uncertainty (variance), which is a statistically sound boundary of the uncertainty in fitness estimation [35]. Notably, recent developments in *multi-task Gaussian process* (MTGP) modeling have shown significant promise in improving the accuracy of surrogates (given comparatively little data) by harnessing the relationships between different but *related* learning tasks [36-37].

It is contended that the discovery and utilization of the correlation between objectives can facilitate the convergence of the *multi-objective EA* (MOEA). In particular, beneficial knowledge can be transferred or shared across objectives to enhance the problem-solving. In [24], a *cross-surrogate assisted memetic algorithm* (CSAMA) was proposed to deal

Jianping Luo is with College of Information Engineering, Shenzhen University, Shenzhen, China, and also with Guangdong Key Laboratory of Intelligent Information Processing and Shenzhen Key Laboratory of Media Security, Shenzhen, China. He currently serves as a Research Scholar at the Computational Intelligence Laboratory (CIL), School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: ljp@szu.edu.cn.

Abhishek Gupta and Yew-Soon Ong are with the Data Science and Artificial Intelligence Research Centre (DSAIR), School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: abhishekg@ntu.edu.sg, asysong@ntu.edu.sg.

Zhenkun Wang is with the School of Computer Science and Engineering, and also with the Air Traffic Management Research Institute (ATMRI), Nanyang Technological University, Singapore. E-mail: wangzk@ntu.edu.sg.

with expensive MOPs. The basic idea was that the construction of the surrogate for one objective could be effectively augmented with information from other related objectives, thereby improving the prediction quality. The associated concept of *multi co-objective evolutionary computation*, defined as a search methodology where information is transferred, exchanged, shared, or reused across objectives, was unveiled. CSAMA was shown to be especially effective under the following conditions: 1) latent correlations exist between different objectives, and 2) the computational cost of different objectives varies widely. For instance, for a 2-objective MOP, where the two objective functions are denoted as f_1 and f_2, respectively, CSAMA facilitates one-way (unidirectional) knowledge transfer from the surrogate model of objective f_1 to the surrogate model of objective f_2. This may be very useful in enhancing the prediction accuracy of the model for f_2, and can also considerably accelerate the optimization search when objective f_1 is cheaper to evaluate than f_2. Notably however, in many practical cases, there may be little direct correlation between objectives.

In this study, we propose a generalization of CSAMA. In particular, we consider the case where all the objectives of an MOP are expensive, with little apparent correlation assumed between them. Accordingly, we incorporate a novel *Gaussian process-based co-sub-Pareto front* (co-subPF) surrogate augmentation strategy within an evolutionary optimizer that we label as GCS-MOE. To elaborate, GCS-MOE considers the MOP to be decomposed into a number of single-objective optimization subproblems in the objective space. The solution of each subproblem is used to approximate a portion or sector of the Pareto front, which we refer to as a *subPF*. The fitness of each subproblem can be expressed as an aggregated scalar value of the different objectives. Notice that, although there may be little correlation between different objectives when considered independently, synergies can indeed exist across subPFs determined by different combinations of the objectives. Beneficial knowledge can therefore be transferred or shared across the subPFs. Hence, multiple *adjacent* scalarized optimization subproblems are expected to be jointly progressed by reusing the similarities among them to enhance the performance. To this end, MTGP surrogate models are built on data generated from the ongoing search to encourage *omnidirectional knowledge transfer across subproblems*.

The major contribution of this paper is in the establishment of an efficient multi-objective algorithm based on the concept of GP-based co-subPF surrogates. The proposed model processes multiple subproblems jointly to improve problem-solving. It facilitates mutual knowledge transfer across subPFs, in a manner that is more general and is also valuable when all the objectives are expensive. Furthermore, we propose a novel criterion for the utility function defined on the surrogate landscape to determine the next candidate solution for evaluation using the actual expensive function. In this regard, a strategy is also presented for effective management of all the evaluated solutions for subsequent model building.

The remainder of this paper is organized as follows. Section II briefly describes MOPs. Section III provides an overview of the Gaussian process and the formulation of its multi-task variant. The GCS-MOE algorithm is introduced in Section IV. A range of experimental studies are showcased in Section V. Finally, the conclusions of the work are drawn in Section VI.

II. Description of Multi-objective Problems

An unconstrained *k*-objective optimization problem can be described as follows:

$$\textit{Minimize } \mathrm{F}(\mathrm{x})=(f_1(\mathrm{x}), f_2(\mathrm{x}),..., f_k(\mathrm{x}))^T,$$
$$\text{Subject to } \mathrm{x} \in \Omega$$

where Ω is the decision space, x is a decision vector, $\mathrm{F}:\Omega \to \mathrm{R}^k$ consists of k real-valued objective functions, and R^k is the objective space. The dominance of solutions in the multi-objective sense is generally defined as:

A vector x_A *is said to dominate* x_B *if* $\mathrm{x}_\mathrm{A}, \mathrm{x}_\mathrm{B} \in \Omega$, $f_i(\mathrm{x}_\mathrm{A}) \le f_i(\mathrm{x}_\mathrm{B})$ $(i \in \{1,2,...,k\})$, *and there is at least one* $j \in \{1,2,...,k\}$, *such that* $f_j(\mathrm{x}_\mathrm{A}) < f_j(\mathrm{x}_\mathrm{B})$, *written as* $\mathrm{x}_\mathrm{A} \prec \mathrm{x}_\mathrm{B}$.

x' *is said to be a non-dominated solution, or a Pareto optimal solution, if* $\mathrm{x}' \in \Omega$ *and there are no other solutions that dominate* x' *in* Ω.

The set of all the Pareto optimal solutions is called the *Pareto set* (*PS*). The objective vectors that correspond to the solutions included in the *PS* are also deemed as non-dominated.

All non-dominated points in the objective function space are collectively known as the *Pareto front (PF).* Formally,

$$PF := \{(f_1(\mathrm{x}), f_2(\mathrm{x}),..., f_k(\mathrm{x})) \mid \mathrm{x} \in PS\}$$

Finding an analytical expression that defines the *PF* of a problem is generally impossible. Thus, the most common way to derive the *PF* is to compute a sufficient number of points in the feasible region and then filter out the non-dominated vectors from them.

III. Gaussian Process and its Multi-task Variant

A. Gaussian Process

Gaussian Process (GP) is also referred to as kriging [38] or DACE stochastic process model [39]. The SAEAs based on GP assume that the objective function is a sample of a GP. The distribution of the objective function value at any untested point can therefore be estimated with the GP model, the hyperparameters of which are trained based on the data collected during the course of the optimization search. As a result, the evaluation for the objective function can be carried out at low cost by the built GP model, as an alternative to the exact expensive function [40-47].

Recently, a number of studies addressing evolutionary optimization of expensive MOPs with GP-based surrogates have been presented. In [48], the authors proposed MOEA/D-EGO for *Expensive Global Optimization* (EGO), which is a variant of the MOEA/D algorithm enhanced with the GP for function approximations. In each iteration of MOEA/D-EGO, the GP is used to make predictions for each subproblem, and the expected improvement (utility function) of these subproblems are optimized simultaneously by using MOEA/D to generate a set of candidate test points that are passed to the exact function evaluation. Further, Keane [49] and Emmerich et al. [50] addressed MOPs by generalizing the *probability of improvement* and the *expected improvement* utility measures. In [51], an EGO algorithm for MOPs has been introduced which adopts the S metric (or hypervolume contribution) to determine which solution to evaluate subsequently. Experimental results showed that the maximization of the hypervolume contribution in real multi-objective optimization can often be a worthwhile alternative for EGO. The well-known ParEGO, which is a direct Pareto extension of EGO, was proposed by Knowles in [52]. A recent survey in [1] provides a comprehensive review on other GP-assisted methods for addressing expensive MOPs.

B. Multi-task GP

Most existing SAEAs are formulated with a single-output predictive model. In other words, given a training set consisting of inputs **x** and vector **y** of corresponding outputs, the mean and variance of the predictive distribution for a new point x* are computed. Typically, a reasonably large number of prior function evaluations will be needed to generate enough data for training a sufficiently accurate model. This is often referred to as the *cold-start problem*. For expensive MOPs in which the number of input observations is naturally expected to be small, multi-task joint learning augments the dataset with a number of different (but related) tasks, such that model parameters can be estimated more confidently than the single-output case [53-54].

Several multi-task GP (MTGP) models, where inter-task knowledge transfer occurs by sharing the kernel function across tasks, have been proposed [55-59]. A model that learns a shared covariance function over input-dependent features and a "free-form" covariance matrix over tasks has been introduced in [36]. The model demonstrates good flexibility when modelling inter-task dependencies and does not require large amounts of data for training.

C. Mathematical Formulation of MTGP

Consider the supervised learning problem given a set of inputs, i.e., the evaluated solutions (training data) $\mathrm{x}_1, \ldots, \mathrm{x}_n$, and corresponding noisy outputs, i.e., the scalar cost $\mathbf{y} = (y_{11}, \ldots, y_{n1}, y_{12}, \ldots, y_{n2}, \ldots, y_{1m}, \ldots, y_{nm})$, where x_i and y_{il} correspond to the ith input and output for task l, respectively, and n is the number of evaluated solutions for task l. The GP approach to this problem is to place a Gaussian prior over the latent functions f_l mapping inputs to outputs. Assuming a zero mean for the outputs, a covariance function that accounts for the dependencies between different tasks, can be defined as [36]:

$$\begin{aligned} &\mathrm{cov}[f_l(\mathrm{x}), f_k(\mathrm{x}')] = K^f_{lk} k^x(\mathrm{x},\mathrm{x}'), \\ &y_{il} \sim N(f_l(\mathrm{x}_i), \sigma_l^2) \end{aligned} \tag{1}$$

where K^f is a positive semi-definite matrix that specifies the inter-task similarities, k^x is a covariance function over inputs, and σ_l^2 is the noise variance for the lth task. To avoid redundancy in the parameterization, k^x can be only a correlation function because the variance can be explained fully by K^f. For $\mathrm{x}=[x_1, \ldots, x_d]$, a commonly used correlation function is the automatic relevance detection (ARD) squared exponential kernel:

$$k^x(\mathrm{x},\mathrm{x}') = \exp(-\sum_{i=1}^{d} \theta_i \mid x_i - x_i^{'} \mid^2), \tag{2}$$

where $\theta_i, i = 1, \ldots, d$, denotes the ith hyperparameter.

Notably, K^f measures the relationship between tasks. This model attempts to learn inter-task dependencies based on the task identities and observed data for each task.

With these definitions, inference can be computed using the standard GP equations for the mean and variance of the predictive distribution, with the covariance function given in Eq. (1).

$$\overline{f_l}(\mathrm{x}^*) = (\mathrm{k}_l^f \otimes \mathrm{k}_*^x)^T C^{-1} \mathbf{y}, \tag{3}$$

$$\mathbf{V}(f_l(\mathrm{x}^*)) = K_{ll}^f k^x(\mathrm{x}^*, \mathrm{x}^*) - (\mathrm{k}_l^f \otimes \mathrm{k}_*^x)^T C^{-1} (\mathrm{k}_l^f \otimes \mathrm{k}_*^x), \tag{4}$$

$$C = K^f \otimes K^x + \boldsymbol{\sigma} \otimes I, \tag{5}$$

where $\otimes$ denotes the Kronecker product, k_l^f represents the lth column of K^f, k_*^x is the vector of covariance between test point x^* and the training points, i.e.,

$$\mathrm{k}_*^x = [k^x(\mathrm{x}^*, \mathrm{x}_1), \ldots, k^x(\mathrm{x}^*, \mathrm{x}_n)]^T,$$

K^x is the matrix of covariance between all pairs of training points, i.e.,

$$K^x = \begin{bmatrix} k^x(\mathrm{x}_1,\mathrm{x}_1) & \cdots & k^x(\mathrm{x}_1,\mathrm{x}_n) \\ \vdots & \ddots & \vdots \\ k^x(\mathrm{x}_n,\mathrm{x}_1) & \cdots & k^x(\mathrm{x}_n,\mathrm{x}_n) \end{bmatrix},$$

$\boldsymbol{\sigma}$ is an $m \times m$ diagonal matrix in which the (l,l)th element is σ_l^2, and C is the overall $mn \times mn$ covariance matrix.

We can learn the parameters θ_i of k^x and matrix K^f by maximizing the log marginal likelihood given as,

$$\Psi(\Theta) = -\frac{1}{2}\log \mid C \mid -\frac{1}{2}\mathbf{y}^T C^{-1}\mathbf{y} - \frac{n \log 2\pi}{2},$$

where $\Theta = \{\boldsymbol{\theta}, K^f\}$ is the set of all hyperparameters. In this study, we use Nystrom approximation of K^x in the marginal likelihood [36] and the PPCA model to learn the K^f [59-60]. These processes are the same as those in literature [36].

The salient feature of the aforementioned model is that it learns inter-task dependencies based on the task identities and observed data for each task, thereby potentially overcoming the cold start problem.

IV. Algorithm Framework

A. Decomposition of MOPs

For an MOP, a Pareto optimal solution can be obtained as an optimal solution of a scalar optimization problem in which the fitness function is an aggregation of all the objectives. In other words, the problem of approximation of the *PF* can be decomposed into a number of single-objective optimization subproblems. Many decomposition-based EAs for handling MOPs have been presented in recent years [61-63]. There are several approaches for converting an MOP into the subproblems. Some of the most commonly-used approaches are the Weighted Sum Approach and Tchebycheff Approach.

For the Tchebycheff approach, which is used throughout this paper, the scalar cost of a subproblem can be defined as follows:

$$\text{minimize} \quad g^{te}(\mathrm{x}\,|\,\lambda^s, \mathrm{z}^*) = \max_{1\le j\le k}\{\lambda_j^s \mid f_j(\mathrm{x}) - z_j^* |\},$$

$$\text{subject to} \quad \mathrm{x}\in\Omega. \tag{6}$$

where λ^s is the weight vector of the subproblem s with $\sum_{j\in\{1,...,k\}}\lambda_j^s = 1$, and $\mathrm{z}^* = (z_1^*,...,z_k^*)^T$ is a reference point approximating the *ideal vector*, i.e., for each $j = 1,...,k$, $z_j^* = \min\{f_j(\mathrm{x}) \mid \mathrm{x}\in\Omega\}$.

Note that, while the weighted sum scalarization approach is intuitively simple, it is not compliant with *concave* Pareto fronts. As no information is typically available beforehand on the shape of the Pareto front, the more flexible Tchebycheff scalarization approach has been preferred in this paper. Nevertheless, it is important to mention that any other scalarization method can directly be incorporated within the general framework proposed herein.

B. Framework

A simple yet efficient GP-based co-subPF surrogate-assisted algorithm (GCS-MOE) is proposed in this study. Similar to other decomposition-based MOEAs, GCS-MOE also decomposes an MOP into several subproblems and optimizes them simultaneously, such that the solution of each of the subproblems can be used to approximate a portion or sector of the *PF* (i.e., a subPF). The Tchebycheff decomposition approach [5] is adopted herein to generate such subproblems. In particular, to generate a uniform distribution and good coverage of the *PF*, the number of subproblems is usually in the hundreds. For expensive MOPs, the optimization for such a large number of subproblems is time consuming. In this work, these subproblems are divided into a number of tasks. As shown in Fig. 1, each task includes a number of adjacent subproblems. In each task, the fitness of the central subproblem is regarded as the representative fitness of this task because the direction vectors of all constitutive subproblems are close to each other and are therefore expected to have similar fitness values. With this, the optimization of subproblems is converted to the optimization of tasks.

Given a set of observations, we aim to learn surrogate models that can predict unobserved response values at several input locations for certain tasks. Similarities are likely to exist between central subproblems of the different tasks, because these subproblems originate from the decomposition of the same MOP. Therefore, information across subPFs can be transferred or shared. Accordingly, a co-subPF surrogate model (i.e., an MTGP model) is built to infer multiple central subproblems jointly, by using the dependencies across subPFs to improve the results.

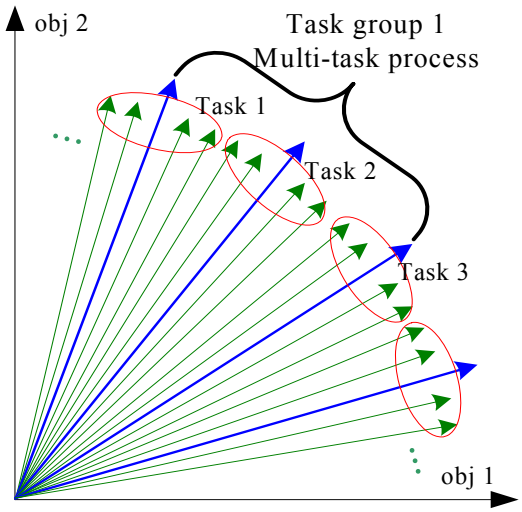


Fig. 1. Division of subproblems into tasks

This framework requires the following parameters.

- N : number of subproblems
- M : total number of tasks
- m : number of tasks for one multi-task GP learning
- g : total number of task groups, $g = \frac{M}{m}$
- ini_size: initial number of evaluated solutions
- $\mathbf{XP} = (\mathrm{xp}^1,...,\mathrm{xp}^{ini\_size})$: initial evaluated solutions
- $\mathbf{X} = (\mathrm{x}^1,...,\mathrm{x}^N)$: population of N solutions, where x^i is the current solution to the ith subproblem
- $\lambda = (\lambda^1,...,\lambda^N)$: uniform distribution of N weight vectors
- **Eva_good_data** = $(\mathrm{Eva\_good\_data}^1,...,\mathrm{Eva\_good\_data}^g)$: good evaluated solutions for each group
- **Eva_common_data** = $(\mathrm{Eva\_common\_data}^1,...,\mathrm{Eva\_common\_data}^g)$: common evaluated solutions for each group
- Max_train_number: maximum number of the evaluated solutions for multi-task GP learning.

Algorithm 1: GCS-MOE

Step 1: Initialization

a) Initialize λ and decompose MOP into N subproblems associated with λ. Set :

Eva_good_data = Eva_common_data $= \varnothing$.

b) Divide the N subproblems into M tasks, and each task consists of $n = \frac{N}{M}$ adjacent subproblems. For each $i = 1,...,M$, the $((i-1)\times n + \left\lfloor \frac{n+1}{2} \right\rfloor)$ th subproblem is regarded as the central subproblem of task i.

c) Divide the M tasks into g task groups. Each task group consists of m adjacent tasks.

d) Generate ini_size= $11\times d-1$ initial evaluated solutions **XP** using the Latin hypercube routine. Evaluate each solution in **XP**. For each $i = 1,...,g$, add **XP** into Eva_common_data^i.

e) Generate the population $\mathbf{X} = (\mathrm{x}^1,...,\mathrm{x}^N)$ using **XP**.

Step 2: New solution generation

Set $Q = \{1,2,...,g\}$

For $i = 1$ **to** g **do**

a) Randomly select one index t from Q.

b) Generate input training data for task group t. (see Algorithm 2)

c) Generate output training data for task group t. For each task j in group t, calculate $\mathbf{y}_j$ for Train_data^t by Eq. (6), where the weight vector of the center subproblem in task j is used. Such that $\mathbf{y} = [\mathbf{y}_1,...,\mathbf{y}_j,...\mathbf{y}_m]$.

d) Build the Gaussian Co-subPF surrogate model. Learn the surrogate model parameters by maximizing the log marginal likelihood:

Model $\leftarrow$ Multi-task_GP_Learning(Train_data^t, $\mathbf{y}$).

e) Evolutionary algorithm search for each task in group t

For each task j in task group t, **do**

i) Generate a new solution to evaluate: x' $\leftarrow$ Evolu_Algorithm(Model).

ii) Evaluate x' using the expensive objective functions.

iii)Update the solutions of the subproblems in task group t using x'. If x' replaces one solution in task group t, set r = TRUE; else set r =FALSE.

iv) Update evaluated data sets for each group, i.e., UpdateEvaData(x', t, r). (see Algorithm 3)

EndFor

f) Delete t from Q.

EndFor

Step 3: Termination

If the stopping criteria are satisfied, then terminate the algorithm and output $\mathbf{X}$. Otherwise, go to Step 2.

C. Initialization

As shown in Algorithm 1, an MOP is decomposed into N subproblems using a set of uniformly distributed weight vectors. The N subproblems are then divided evenly into M tasks, and each task consists of $n = \frac{N}{M}$ adjacent subproblems. For each i, $i = 1,...,M$, the $((i-1)\times n + \left\lfloor \frac{n+1}{2} \right\rfloor)$th subproblem is regarded as the central subproblem of task i, and the other $n-1$ nearest neighbor subproblems are then chosen as the subproblems of task i according to the Euclidean distance between the central subproblem and the other subproblems. In each task, the fitness of the central subproblem is regarded as the representative fitness of this task. Next, these tasks are allocated to g task groups. Each task group consists of the same number of adjacent tasks. See Fig.1 for an illustration of subproblems, tasks, and task groups. The initial evaluated solutions are generated using a Latin hypercube routine following a description in Numerical Recipes [64]. The number of initial test points is set to $11\times d-1$ in all the test instances, where d is the decision space dimensionality of the function to be optimized, and the setting is similar to that in literature [52]. These solutions (denoted as **XP**) are evaluated using the expensive objective function and used to build the surrogate model. For each subproblem, a solution is selected from the initially evaluated set of solutions (based on their scalar cost) to represent the best solution found so far for the subproblem. The number of initially evaluated solutions is less than the number of subproblems, so each initially evaluated solution can be selected more than once. These solutions of subproblems constitute the population $\mathbf{X}$.

Algorithm 2: Generate input training data for task group t

Set Train_data^t = $\varnothing$;

Train_data^t $\leftarrow$ Eva_good_data^t.

If (sizeof(Train_data^t)< Max_train_number)

If(Max_train_number - sizeof(Train_data^t) $\geq$ sizeof(Eva_common_data^t)

Add Eva_common_data^t into Train_data^t.

Else

Randomly select Max_train_number - sizeof(Train_data^t) solutions from Eva_common_data^t, and add these solutions into Train_data^t.

EndIf

EndIf

D. Using the GP-based Co-subPF Surrogate Model

In the GCS-MOE, a co-subPF MTGP surrogate model is built for each task group. As shown in Fig.1, the MOP is converted into a number of tasks to be optimized. The fitness of each task (i.e., the fitness of the central subproblem in this task) can be obtained by combining the objective values using the appropriate scalarization method. Similarities are likely to exist across subPFs in one task group since all the constitutive subproblems are adjacent to each other in the objective space. Thus, useful information can be shared among central subproblems via the matrix K^f, defined in Eq. (1), to improve the accuracy of the multi-task learning GP model in one task group. As shown in Step 2-c of Algorithm 1, to build the surrogate model more accurately, the modeling of tasks in one group are processed jointly, i.e., the input training data (evaluated solutions) and corresponding outputs of all the tasks in one group are used simultaneously. The approach is different from the standard GP where the outputs are only considered from one task (or subproblem) at a time. It is clear that the proposed surrogate model augments the observed dataset with a number of related tasks, such that model parameters can be estimated more confidently. The learning of the surrogate model parameters by maximizing the log marginal likelihood function is introduced in Section III (Step 2-d of Algorithm 1). Once the surrogate model is learned for one task group, it is used for prediction purposes of all tasks within this group (Step 2-e of Algorithm 1). The utility function of each task (i.e., the merit value predicted by the surrogate model for an unknown candidate solution) is optimized with an evolutionary scheme to generate the next candidate test solution (discussed in Section IV-E). The solution is then evaluated using the expensive objective functions, thereafter updating the population $\mathbf{X}$ and the evaluated data sets.

Algorithm 3: Update the evaluated data sets for each task group, UpdateEvaData(x', t, r).

If(r = **True**)
  If(sizeof(Eva_good_data^t < Max_train_number)
    Add x' into Eva_good_data^t .
  Else
    A randomly selected solution is replaced by x' in Eva_good_data^t .
  EndIf
Else
  If(sizeof(Eva_common_data^t < Max_train_number)
    Add x' into Eva_common_data^t .
  Else
    A randomly selected solution is replaced by x' in Eva_common_data^t .
  EndIf
EndIf
For each group $i, i \in \{1,...,g\}, i \neq t$
  If(sizeof(Eva_common_data^i) Max_train_number)
    Add x' into Eva_common_data^i .
  Else
    A randomly selected solution is replaced by x' in Eva_common_data^i .
  EndIf
EndFor

E. Selecting a candidate solution for function evaluation

After the surrogate model is built, to find the best solution to visit next for one task, an evolutionary optimizer is applied to search the decision parameter space globally. As shown in Step 2-e-i of Algorithm 1, in our framework, any meta-heuristic, such as a genetic algorithm, particle swarm optimization, etc., can be used to generate the candidate solution for exact function evaluation. In this study, a real-coded genetic algorithm-based approach similar to the literature [52] is applied, and it searches for the solution that *minimizes the negated figure of merit value* of the solution (i.e., utility function) as predicted by the surrogate model. The estimation of the merit value of a solution in the search process is detailed in Section IV-D.

Algorithm 4: Utility function for a solution x*
Input: $f_l(\mathrm{x}^*) \sim N(\overline{f_l}(\mathrm{x}^*), \mathbf{V}(f_l(\mathrm{x}^*)))$, $f_l^{\min}$

If ($\overline{f_l}(x^*) > f_l^{\min}$)
  $f_{Utility} = E[I(x^*)] - P$
Else
  $f_{Utility}$ is set to a value according to Eq. (11)
EndIf
Return $f_{Utility}$

F. Utility function

In the literature, there are several criteria that have been proposed to determine the figure of merit for the exact evaluation of the objective functions at a new point. The first is the expected improvement (EI) criterion, which has been widely used. For an untested solution x*, suppose that $f_l(\mathrm{x}^*)$ is a predictive distribution model for an objective function of the lth task, i.e., $f_l(\mathrm{x}^*) \sim N(\overline{f_l}(\mathrm{x}^*), \mathbf{V}(f_l(\mathrm{x}^*)))$, and $f_l^{\min}$ is the current best function value among all evaluated solutions. The expected improvement can then be written as

$$E[I(\mathrm{x}^*)] = E[\max\{f_l^{\min} - f_l(\mathrm{x}^*), 0\}]. \quad (7)$$

Applying integration by parts, Eq. (7) can be expressed as

$$E[I(\mathrm{x}^*)] = (f_l^{\min} - \overline{f_l}(\mathrm{x}^*))\Phi\left(\frac{f_l^{\min} - \overline{f_l}(\mathrm{x}^*)}{\sqrt{\mathbf{V}(f_l(\mathrm{x}^*))}}\right) + \sqrt{\mathbf{V}(f_l(\mathrm{x}^*))}\phi\left(\frac{f_l^{\min} - \overline{f_l}(\mathrm{x}^*)}{\sqrt{\mathbf{V}(f_l(\mathrm{x}^*))}}\right), \quad (8)$$

where $\Phi(.)$ and $\phi(.)$ are the standard normal distribution and density function, respectively.

This utility function accounts for both the predicted value of solutions as well as the uncertainty in the prediction to find the one that has the greatest potential to improve the minimum cost.

Cox and John [65] developed a similar method by proposing the criterion

$$F(\mathrm{x}^*) = -\left(\overline{f_l}(\mathrm{x}^*) - \omega\sqrt{\mathbf{V}(f_l(\mathrm{x}^*))}\right),\ \omega > 0. \quad (9)$$

The two components of $F(\mathrm{x}^*)$ automatically balance exploitation and exploration. $F(\mathrm{x}^*)$ is better (larger) for a minimization problem when the mean predicted value is small and/or the standard error is large (i.e., the prediction is very uncertain). Fig. 2 shows this situation, where x^1, x^2, $\mathrm{x}^3$and x^4 are the evaluated points, and point x^c is likely to be selected for evaluation because it has a small predicted mean value and a large variance value.

Another popular criterion is the probability of improvement (POI)[66] defined as

$$F(\mathrm{x}^*) = \Phi\left(\frac{f_l^{\min} - E_n - \overline{f}_l(\mathrm{x}^*)}{\sqrt{\mathbf{V}(f_l(\mathrm{x}^*)}}\right). \qquad (10)$$

The location of the best guess is obtained by finding the point with the maximum probability that x^* is less than $f_l^{\min}$ by at least a positive constant E_n . However, the nature of the probabilities generated by the model dictates the method to guess in areas with few evaluations of the function (where the variance is high) in an attempt to cover the entire space. One can refer to reference [67] for discussions about more criteria.

In this study, we develop a new method to address the issue. For expensive MOPs, function evaluations are precious. The key idea of our approach is that for solutions whose predicted mean value is larger than the current best value, exploration is considered to be of more interest. By contrast, for solutions whose predicted mean value is less than the current best value, exploitation is considered to be of more interest. Therefore, as shown in Algorithm 4 (where P is a large positive constant), when the predicted mean value of a solution is larger than the current best value, the *expected improvement* criterion is used; otherwise, the criterion used to determine the figure of merit for evaluation is defined as

$$F(\mathrm{x}^*) = -\left(\overline{f}_l(\mathrm{x}^*) + \gamma\sqrt{\mathbf{V}(f_l(\mathrm{x}^*))}\right), \ \gamma>0. \qquad (11)$$

In this situation (i.e., the predicted mean value of a solution is less than the current best value), our method is different from Cox and John's method. Our method places more emphasis on the probability that x^* is less than $f_l^{\min}$ because $\gamma>0$. This is reasonable because x^* is evaluated only if it is likely to be truly less than $f_l^{\min}$, thereby avoiding wasting an expensive evaluation. The factor γ can be adjusted to balance the global exploration and local search. The smaller the γ is, the stronger the global search will be. In this paper, we tried different γ values and through trial and error, γ is set to be 0.5 throughout. Fig. 2 shows that point x^b is likely to be selected for evaluation because x^b has a minimum value according to our proposed criterion curve. Although the predicted mean values of x^b and x^c are the same in Fig. 2, x^b is better than x^c according to our criterion. This result is reasonable because we place more emphasis on obtaining a solution that is predicted to be better with greater certainty (i.e., yield a lower cost as compared to $f^{\min}$).

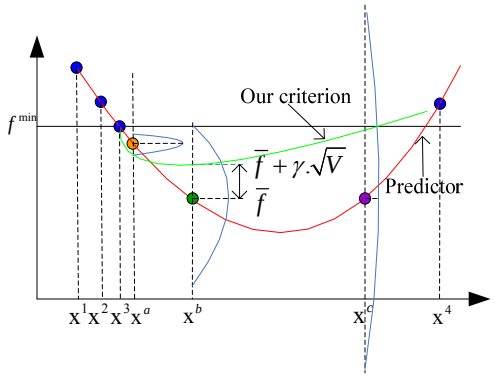


Fig. 2. Uncertainty about the utility function's value at several untested points

In our algorithm, the solution that minimizes the utility function with respect to Algorithm 4 is deduced using an evolutionary approach. This solution becomes the next point to be evaluated on the expensive objective functions.

G. *Management of Evaluated Solutions for Model Training*

In many GP models, some solutions are selected randomly from all the evaluated solutions to update the surrogate model. In our framework, each task group corresponds to a surrogate model, i.e., the evaluated solution sets used to learn the model parameters are different for each model. As shown in Algorithm 3, each task group possesses a good evaluated solution set (eva_good_data) and a common evaluated solution set (eva_common_data). This approach will likely lead to more effective predictions because different task groups have different attributes. As shown in Step 2-e-iii of Algorithm 1, for each task group, after the candidate solution x' is evaluated using the expensive objective function, it will be used to update the population $\mathbf{X}$ and the evaluated data sets. If the evaluated solution can be accepted in population $\mathbf{X}$ (i.e., replace the solution of a subproblem in one task group), the solution can be assigned to the good evaluated data set and has a high priority to be used to update the surrogate model in the next iteration. Otherwise, the evaluated solution is assigned to the common evaluated solution set. The common evaluated solution set can only be used when the number of good evaluated solutions is less than the maximum number of solutions for model training.

H. *Comparison with MOEA/D-M2M, ParEGO and MOEA/D-EGO*

MOEA/D-M2M [68] uses some strategies to decompose an MOP into several multi-objective optimization subproblems and also solves them in a collaborative manner. However, the algorithm is not tailored for expensive MOPs, and the collaborative approach is not based on the MTGP model. ParEGO builds a GP model for merely one aggregation function (i.e., subproblem) at a time at each iteration. Therefore, it does not make full use of the shared knowledge among aggregation functions, and thus, the built GP may not be sufficiently accurate. In MOEA/D-EGO, a separate GP model is built for each objective function. Then, the predictive values are approximated for each subproblem based on the built GP models. At each generation, MOEA/D is used for maximizing the expected improvement metric values of all the subproblems. However, the errors (inaccuracies) of one GP

model built for one objective can be transmitted to the estimated values of all the subproblems using this approach. In our framework, a multi-task GP model for each task group is built. The model considers several scalarized tasks to be combined to form one task group. Beneficial knowledge can therefore be seamlessly transferred or shared across subPFs approximated by these subproblems. Hence, the GP model parameters can be estimated more confidently.

V. Experimental Studies and Discussion

We focus on two- and three-objective optimization problems in this work. GCS-MOE is compared with K-RVEA [26], SAMO [69], MOEA/D-EGO [48], ParEGO [52] and NSGA-II to analyze the performance of GCS-MOE and understand its behavior. K-RVEA is run in the PlatEMO platform [70], and the codes of the other compared algorithms are downloaded from the websites or links given by their authors[1]. NSGA-II is also selected as a comparative algorithm because it demonstrates good performance on two- and three-objective problems.

A. Test Problems

1) **Benchmark test problems** CEC'09 benchmark problems [71] are selected as the two- and three-objective instances. In addition, ZDT [72] instances are selected as two-objective problems. All these test instances, with various characteristics, are minimization problems. The number of decision variables is set to 3 for CEC'09 UF1–UF7 instances, 5 for CEC'09 UF8–UF10 instances, and 8 for ZDT instances.

2) **Real-world engineering application problems** The Welded Beam problem (WBP) [69] and the Tool Spindle Design problem (TSDP) [69] are tested in our experiments. They are both well-studied engineering case studies. WBP is a bi-objective optimization problem with four design variables (thickness of the beam, width of the beam, length of the weld, and weld thickness) with the objectives being cost and end deflection. There are four inequality constraints, relating to geometry, weld shear stress, beam buckling load, and beam bending stress. TSDP is also a bi-objective minimization problem having four design variables. The objectives of the problem are the volume of the spindle and static displacement. There are two inequality constraints on designer's proportion requirements and one inequality constraint on maximal radial runout of the spindle nose.

B. Performance Metrics

Several performance metrics in multi-objective optimization have been discussed in the literature [73]. In our experiments, the following performance indexes are used.

Hypervolume Indicator (HV-metric) [74]**:**

Let $y^* = (y_1^*, ..., y_k^*)$ be a point in the objective space, which is dominated by all Pareto-optimal objective vectors. Hypervolume is a metric that evaluates the volume of the space dominated by an approximation set, relative to the reference point y^* in the objective space. The formula is given in Eq. (12), where j is the number of non-dominated solutions in the approximation set, and v_i is the Hypervolume contribution of the *i-th* solution relative to the reference point.

$$Hypevolume = volume\left(\bigcup_{i=1}^{j} v_i\right) \tag{12}$$

Inverted Generational Distance (IGD-metric) [75]:

We let P^* be a set of well representative points along the *PF* (in the objective space). We let A be an approximation to the *PF*. The average distance from P^* to A is defined as follows:

$$IGD(A, P^*) = \frac{\sum_{F \in P^*} d(F, A)}{|P^*|}, \tag{13}$$

where $d(F, A)$ is the minimum Euclidean distance between F and the points in A. If $|P^*|$ is sufficiently large to represent the *PF* well, $IGD(A, P^*)$ could measure both the diversity and convergence of A to a certain extent. A must be very close to the *PF*, and must not miss any part of the entire *PF*, to obtain a low value of $IGD(A, P^*)$.

The aforementioned metrics are computed by jMetal [76].

C. Parameter Settings

The maximum number of function evaluations for all the algorithms is set to 200. Each instance is tested by each algorithm over 10 independent runs.

For GCS-MOE, the creation of N weight vectors $(\lambda^1, ..., \lambda^N)$ is controlled by a positive integer parameter H, which specifies the granularity or resolution of the weight vectors, as in [5]. Each individual weight takes a value from

$$\{\frac{0}{H}, \frac{1}{H}, ..., \frac{H}{H}\}.$$

Thus, the number of weight vectors can be calculated as [5]

$$N = C_{H+k-1}^{k-1}, \tag{14}$$

where k is the number of objectives. H is set to 149 for two-objective problems and 20 for three-objective problems. Therefore, N is 150 for two-objective problems and 230 for three-objective problems. The total number (M) of tasks is important in our algorithm. M is 15 for two-objective problems and 23 for three-objective problems. Therefore, N subproblems are divided into M tasks, where each task contains 10 adjacent subproblems. The number of tasks (m) for one multi-task GP learning is set to be 3. More discussions about parameters M and m are in the section V-F. The number of evaluated solutions considered for model building is set to a maximum of 80 in our experiments, because the computational overhead increases cubically with the size of the corresponding covariance matrix. For K-RVEA, SAMO, ParEGO and MOEA/D-EGO, all the parameter settings are the same as those in the literature. For NSGA-II, the population size is reduced to 20, to maximize the performance in the short-run experiments. Other parameter settings are similar to those in [52].

D. Comparison experiments on benchmark problems

Tables I and II list the mean and standard deviation values of

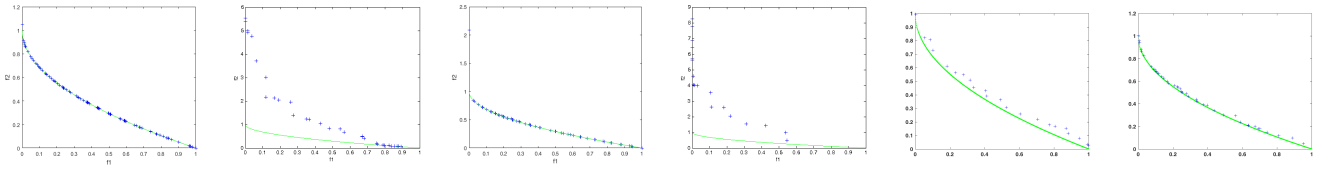
[1]SAMO: http://www.mdolab.net/Ray/Research-Data/SAMO_JMD.zip
MOEA/D-EGO: http://dces.essex.ac.uk/staff/zhang/IntrotoResearch/ExpensiveOptimization.htm
ParEGO: http://dbk.ch.umist.ac.uk/knowles/parego/
NSGA-II: http://www.egr.msu.edu/~kdeb/codes.shtml
PlatEMO: http://bimk.ahu.edu.cn/index.php?s=/index/software/index.html

TABLE I
MEAN AND STANDARD DEVIATION VALUES OF IGD OBTAINED BY THE ALGORITHMS.

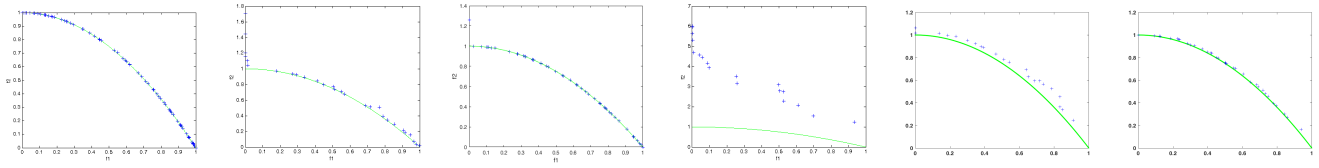
Ins.	GCS-MOE	ParEGO		MOEA/D-EGO		NSGA-II		SAMO		K-RVEA	
ZDT1	**0.01325(0.00293)**	0.11230(0.06765)	↑	0.01503(0.00068)	~	0.85724(0.29670)	↑	0.07582(0.02408)	↑	0.02877(0.00920)	↑
ZDT2	**0.00792(0.00063)**	0.10932(0.06396)	↑	0.01525(0.00272)	~	1.49606(0.58739)	↑	0.08898(0.03343)	↑	0.04648(0.02779)	↑
ZDT3	**0.01206(0.00126)**	0.35032(0.09552)	↑	0.06541(0.00986)	↑	1.09520(0.25585)	↑	0.21676(0.09550)	↑	0.03696(0.01186)	↑
ZDT4	1233.95204(388.51472)	39.14445(7.30210)	↓	53.82244(15.77437)	↓	800.89210(417.48926)	~	42.60012(12.36217)	↓	**25.01193(7.76258)**	↓
ZDT6	**0.02365(0.01358)**	0.79778(0.10725)	↑	0.05799(0.01109)	↑	6.13114(0.58605)	↑	1.24964(0.13651)	↑	2.12807(0.46917)	↑
UF1	**0.02625(0.00755)**	0.05947(0.02181)	↑	0.06249(0.03698)	↑	0.07282(0.01130)	↑	0.13539(0.06825)	↑	0.03950(0.01337)	↑
UF2	**0.02538(0.00519)**	0.03694(0.00794)	↑	0.02932(0.00127)	~	0.04481(0.00545)	↑	0.07039(0.02478)	↑	0.03402(0.01107)	~
UF3	**0.30028(0.15409)**	0.40941(0.18354)	~	0.37680(0.00341)	~	0.70364(0.33857)	↑	0.55279(0.16370)	~	0.56284(0.35552)	↑
UF4	**0.04238(0.00155)**	0.06083(0.00600)	↑	0.05037(0.00273)	~	0.07804(0.00888)	↑	0.09624(0.00731)	↑	0.06949(0.02046)	~
UF5	**0.24214(0.07070)**	0.33065(0.10330)	↑	0.82013(0.41080)	↑	0.62764(0.29619)	↑	1.27160(0.41311)	↑	0.75542(0.34913)	↑
UF6	**0.01859(0.01058)**	0.30368(0.06210)	↑	0.56596(0.16924)	↑	0.12482(0.04979)	↑	1.72130(0.60145)	↑	1.28608(0.66696)	↑
UF7	**0.04526(0.00623)**	0.07534(0.05045)	↑	0.07999(0.01933)	↑	0.14890(0.07119)	↑	0.23967(0.06393)	↑	0.11683(0.08174)	↑
UF8	**0.11258(0.04505)**	0.22472(0.04757)	↑	0.12508(0.00528)	~	0.90236(0.77881)	↑	0.40166(0.10030)	↑	0.12571(0.01177)	~
UF9	0.15283(0.02385)	0.20260(0.03131)	↑	0.16880(0.09537)	~	0.74322(0.50683)	↑	0.42848(0.20722)	↑	**0.12248(0.01886)**	↓
UF10	**1.36080(0.50432)**	1.69478(0.60456)	↑	1.64695(0.96512)	↑	2.85992(0.85683)	↑	3.05480(1.13977)	↑	1.60132(0.52306)	↑

TABLE II
MEAN AND STANDARD DEVIATION VALUES OF HV OBTAINED BY THE ALGORITHMS.

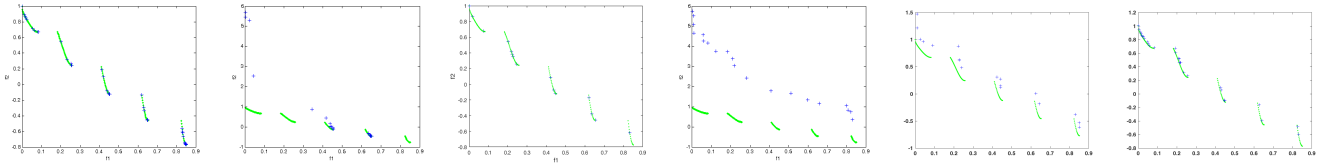
Ins.	GCS-MOE	ParEGO		MOEA/D-EGO		NSGA-II		SAMO		K-RVEA	
ZDT1	**0.64829(0.00301)**	0.21441(0.07819)	↑	0.61022(0.00141)	↑	0.03966(0.07416)	↑	0.53951(0.03675)	↑	0.62503(0.00806)	↑
ZDT2	**0.30256(0.00166)**	0.21742(0.07189)	↑	0.28850(0.00604)	↑	0.00000(0.00000)	↑	0.21404(0.03215)	↑	0.27403(0.02660)	↑
ZDT3	0.76302(0.00295)	0.36561(0.11763)	↑	0.69472(0.00986)	↑	0.02500(0.03824)	↑	0.56618(0.11301)	↑	**0.76826(0.00430)**	~
ZDT4	0.00000(0.00000)	0.00000(0.00000)	~	0.00000(0.00000)	~	0.00000(0.00000)	~	0.00000(0.00000)	~	0.00000(0.00000)	~
ZDT6	**0.18027(0.04438)**	0.00340(0.00488)	↑	0.08832(0.04908)	↑	0.00000(0.00000)	↑	0.00000(0.00000)	↑	0.00000(0.00000)	↑
UF1	**0.60223(0.01739)**	0.56029(0.03408)	↑	0.54791(0.06822)	↑	0.53263(0.01719)	↑	0.44362(0.10077)	↑	0.59632(0.02084)	~
UF2	**0.61985(0.00725)**	0.60904(0.00996)	↑	0.60190(0.00194)	~	0.58488(0.01062)	↑	0.54026(0.04746)	↑	0.60181(0.02482)	↑
UF3	**0.19335(0.16524)**	0.18370(0.10849)	~	0.00883(0.01988)	↑	0.07483(0.09538)	↑	0.07592(0.07214)	↑	0.13693(0.12162)	~
UF4	**0.26862(0.00296)**	0.22353(0.01077)	↑	0.25229(0.00625)	~	0.20799(0.01162)	↑	0.18659(0.01622)	↑	0.22366(0.02782)	↑
UF5	**0.19880(0.06729)**	0.13916(0.08531)	↑	0.01773(0.03550)	↑	0.06240(0.06771)	↑	0.00150(0.00449)	↑	0.04432(0.05239)	↑
UF6	**0.38582(0.02250)**	0.08927(0.04255)	↑	0.01524(0.04571)	↑	0.29335(0.07829)	↑	0.00000(0.00000)	↑	0.01340(0.03162)	↑
UF7	**0.40958(0.01436)**	0.39892(0.04381)	~	0.39087(0.03423)	↑	0.31192(0.06236)	↑	0.23277(0.06138)	↑	0.36123(0.07008)	↑
UF8	**0.21059(0.02541)**	0.15813(0.01926)	↑	0.20389(0.03605)	~	0.02736(0.03983)	↑	0.02293(0.01798)	↑	0.18038(0.02991)	↑
UF9	0.48952(0.08960)	0.31195(0.04108)	↑	0.51293(0.09819)	~	0.13490(0.14019)	↑	0.19052(0.09856)	↑	**0.52467(0.02874)**	↓
UF10	**0.00238(0.00084)**	0.00123(0.00369)	↑	0.00179(0.00785)	↑	0.00000(0.00000)	↑	0.00000(0.00000)	↑	0.00000(0.00000)	↑



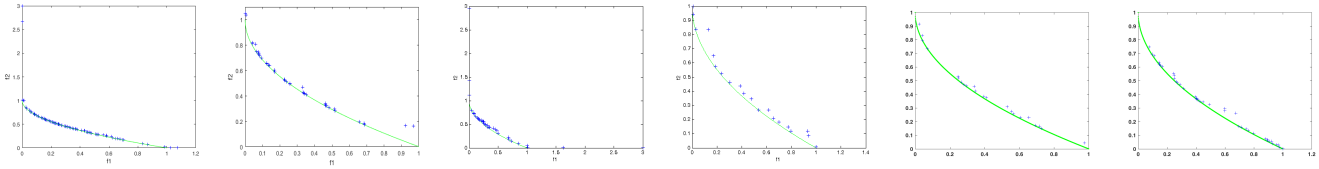
(a) ZDT1 (GCS-MOE) (b) ZDT1 (ParEGO) (c) ZDT1 (MOEA/D-EGO) (d) ZDT1 (NSGA-II) (e) ZDT1 (SAMO) (f) ZDT1 (K-RVEA)



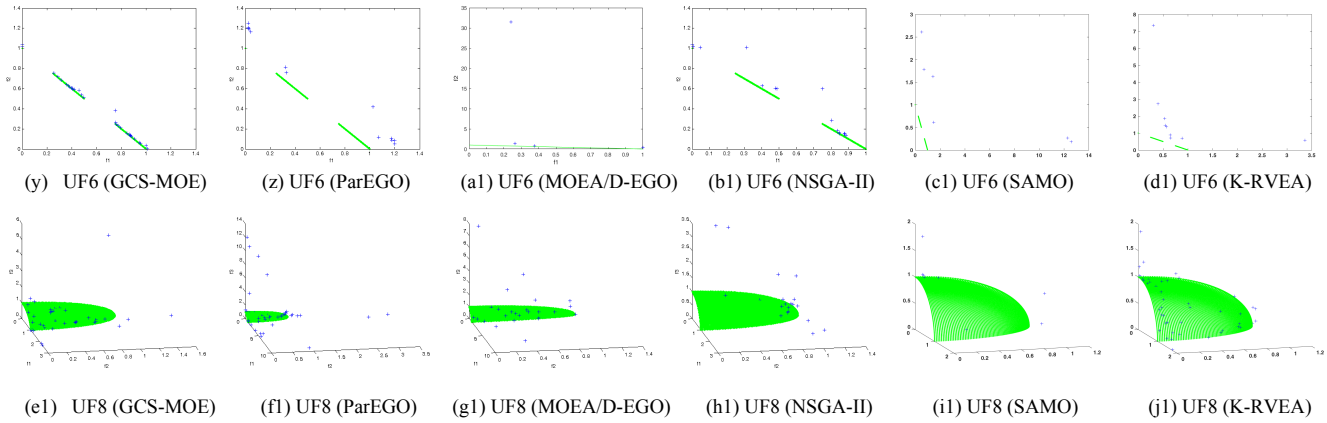
(g) ZDT2 (GCS-MOE) (h) ZDT2 (ParEGO) (i) ZDT2 (MOEA/D-EGO) (j) ZDT2 (NSGA-II) (k) ZDT2 (SAMO) (l) ZDT2 (K-RVEA)



(m) ZDT3 (GCS-MOE) (n) ZDT3 (ParEGO) (o) ZDT3 (MOEA/D-EGO) (p) ZDT3 (NSGA-II) (q) ZDT3 (SAMO) (r) ZDT3 (K-RVEA)



(s) UF2 (GCS-MOE) (t) UF2 (ParEGO) (u) UF2 (MOEA/D-EGO) (v) UF2 (NSGA-II) (w) UF2 (SAMO) (x) UF2 (K-RVEA)



(y) UF6 (GCS-MOE) (z) UF6 (ParEGO) (a1) UF6 (MOEA/D-EGO) (b1) UF6 (NSGA-II) (c1) UF6 (SAMO) (d1) UF6 (K-RVEA)

(e1) UF8 (GCS-MOE) (f1) UF8 (ParEGO) (g1) UF8 (MOEA/D-EGO) (h1) UF8 (NSGA-II) (i1) UF8 (SAMO) (j1) UF8 (K-RVEA)

Fig. 3. Plots of the final approximations in the objective space on ZDT1, ZDT2, ZDT3, UF2, UF6, and UF8 instances

the HV and IGD metrics in all the test problems, for comparison of the six algorithms. At the same time, we apply the Wilcoxon nonparametric statistical test on the algorithms. In the tables, "↑" "↓" and "~" denote that GCS-MOE is better, worse, or equal than the algorithm used for comparison with a 95% significance level. Fig. 3 shows the distribution of the approximation set in the objective space, obtained in the run with the lowest IGD value of each algorithm, in several test instances. Tables I and II show that GCS-MOE is almost always the best algorithm for all these problems. GCS-MOE has the best HV and IGD values in most of the problems, indicating that the algorithm exhibits stable performance in these test instances. K-RVEA and MOEA/D-EGO perform better than ParEGO, SAMO and NSGA-II in the ZDT instances. However, GCS-MOE performs better than or similarly to K-RVEA and MOEA/D-EGO in these instances, with the sole exception of ZDT4.

For example, for the ZDT instances in Fig. 3, the evolved solutions of NSGA-II are far from the true *PF*. ParEGO performs better than NSGA-II. However, particularly for the case of ZDT3, only a small part of the solutions can converge to the true *PF*. Moreover, these solutions are found to be highly concentrated (lacking diversity) in the objective space. By contrast, the figure of the *PF* approximation of GCS-MOE shows that most of the solutions of GCS-MOE can converge close to the true *PF*. The results indicate that the GCS-MOE is indeed superior among the considered algorithms to address the benchmark problem. What is more, similar outcomes can be observed for UF instances as well.

In our experiments, GCS-MOE, K-RVEA, SAMO, ParEGO and MOEA/D-EGO all use a GP or MTGP model as a surrogate. The computation time of different algorithms for training the models is often the concern of a practitioner. The training times can vary significantly depending on the specific implementation and the number of input samples used for model building. Accordingly, a comparison is presented in this section. In GCS-MOE, ParEGO and MOEA/D-EGO, the number of input samples for model building is set to a maximum of 80. SAMO builds the surrogate models for each constraint and objective function separately using different types of approximation methods: RBF, kriging, MLP, and RSM [69], and selects the best surrogate in each generation. The training times obtained for different algorithms on the 3-objective UF8 example are shown in Table III. We can observe that the running time of GCS-MOE is not small. One reason for this is that the covariance matrix (C in Eq.(5)) is very large in the multi-task learning case due to the Kronecker product. Nevertheless, the running time of the training model in GCS-MOE is less than SAMO and MOEA/D-EGO. On the other hand, K-RVEA is found to have the least training time among the compared algorithms. Note that the model building of GCS-MOE and SAMO is implemented in MATLAB, MOEA/D-EGO is in Java, and ParEGO in C++. Therefore, the absolute time used by the different algorithms may not be directly comparable and serves as a reference only.

TABLE III
AVERAGE TRAINING TIME FOR DIFFERENT ALGORITHMS ON UF8

Algorithm	GCS-MOE	ParEGO	MOEA/D-EGO	SAMO	K-RVEA
Time (s)	5.8	0.9	9.2	9.5	0.3

E. Comparison on engineering optimization problems

WBP and TSDP problems are constrained MOPs drawn from engineering applications. To solve these two problems in our experiments, when the generated solution doesn't satisfy the constraints, a large positive integer (penalty term) is added to each objective function in GCS-MOE, ParEGO, and MOEA/D-EGO. Since these two problems are not included in PlatEMO, K-RVEA is not added to the comparison. For each problem, the non-dominated solutions across all independent optimization runs of every algorithm are combined and regarded as the true *PF*. To effectively calculate the HV values, the first objective value of these two problems is divided by 60 and 1,600,000 (as a form of scaling), respectively. Tables IV lists the mean, best, worst, and standard deviation values of the HV and IGD metrics in the WBP and TSDP problems for comparison of the four surrogate-assisted algorithms. To make the comparison results simpler to decipher, the best results for each problem are bold and underlined, while the second best ones are only bold. GCS-MOE is among the top 2 of all compared algorithms in this study. However, it can be observed that SAMO performs slightly better than GCS-MOE on average. The reason for it is that, besides the objective functions, SAMO also builds an explicit surrogate model for each constraint, thereby suggesting superior performance in constraint handling. Nevertheless, the mean, best, and worst values of GCS-MOE are better than or similar to those of ParEGO and MOEA/D-EGO. Therefore, it is considered that GCS-MOE can indeed effectively handle real-world engineering optimization problems as well.

TABLE IV

MEAN, BEST, WORST, AND STANDARD DEVIATION VALUES OF IGD AND HV OBTAINED BY THE ALGORITHMS.

Problems		HV				IGD			
		SAMO	GCS-MOE	ParEGO	MOEA/D-EGO	SAMO	GCS-MOE	ParEGO	MOEA/D-EGO
Welded Beam	Mean	**0.87649**	**0.87253**	0.83626	0.85305	**0.01249**	**0.01689**	0.02836	0.01892
	Best	**0.92229**	**0.91304**	0.85103	0.90651	**0.00286**	**0.00627**	0.01254	0.00937
	Worst	**0.82715**	**0.79606**	0.78549	0.78252	**0.02542**	**0.04403**	0.05540	0.04963
	Std.	0.03032	0.04440	0.02017	0.04216	0.00837	0.01570	0.00637	0.00940
Tool Spindle Design	Mean	**0.47041**	**0.38022**	0	0.36887	**0.00513**	**0.02468**	1.00156	0.02495
	Best	**0.50847**	**0.45776**	0	0.44503	**0.00114**	0.00429	0.85655	**0.00410**
	Worst	**0.37303**	**0.32836**	0	0.32106	**0.02335**	**0.04271**	1.23521	0.06897
	Std.	0.04106	0.04754	0	0.03951	0.00633	0.01365	0.11568	0.00987

TABLE V

MEAN AND STANDARD DEVIATION VALUES OF IGD AND HV OBTAINED USING DIFFERENT UTILITY FUNCTIONS.

	IGD				HV			
Ins.	EI	Cox and John	POI	Proposed criterion	EI	Cox and John	POI	Proposed criterion
ZDT1	0.01320(0.00085)	0.10298(0.08937)	0.06497(0.01032)	**0.01325(0.00293)**	0.62951(0.00048)	0.47716(0.16083)	0.54616(0.02108)	**0.64829(0.00301)**
ZDT2	0.01109(0.00168)	0.08261(0.04882)	0.04905(0.00496)	**0.00792(0.00063)**	0.28983(0.01168)	0.15878(0.10330)	0.23146(0.01234)	**0.30256(0.00166)**
ZDT3	0.03095(0.04280)	0.07588(0.04614)	0.07957(0.01871)	**0.01206(0.00126)**	0.71027(0.04196)	0.67469(0.08175)	0.68647(0.01756)	**0.76302(0.00295)**
ZDT4	**1160.30285(480.20569)**	2018.62413(489.25033)	1332.06180(576.73632)	1233.95204(388.51472)	0.00000(0.00000)	0.00000(0.00000)	0.00000(0.00000)	0.00000(0.00000)
ZDT6	0.05016(0.00607)	0.05753(0.01330)	0.04448(0.02288)	**0.02365(0.01358)**	0.08698(0.01367)	0.06244(0.04644)	0.13397(0.10812)	**0.18027(0.04438)**
UF1	0.05485(0.01390)	0.12446(0.01585)	0.06516(0.00600)	**0.02625(0.00755)**	0.57446(0.03258)	0.42588(0.03478)	0.54814(0.00744)	**0.60223(0.01739)**
UF2	0.03125(0.00519)	0.12507(0.05100)	0.07615(0.03668)	**0.02538(0.00519)**	0.60193(0.01181)	0.45967(0.06588)	0.52076(0.06633)	**0.61985(0.00725)**
UF3	0.34820(0.00285)	0.35325(0.00410)	**0.22984(0.00328)**	0.30028(0.15409)	0.02408(0.00020)	0.00000(0.00000)	**0.27800(0.01716)**	**0.19335(0.16524)**
UF4	0.04693(0.00571)	0.06961(0.01507)	0.07881(0.00851)	**0.04238(0.00155)**	0.26170(0.00445)	0.21276(0.02824)	0.19240(0.02210)	**0.25862(0.00296)**
UF5	0.30215(0.09314)	1.79322(0.46263)	0.42887(0.08685)	**0.24214(0.07070)**	0.14310(0.04107)	0.00000(0.00000)	0.08948(0.07686)	**0.19880(0.06729)**
UF6	0.04768(0.00449)	0.07577(0.02451)	0.04715(0.01159)	**0.01859(0.01058)**	0.36198(0.02093)	0.30534(0.03783)	0.33865(0.01361)	**0.38582(0.02250)**
UF7	0.07135(0.00496)	0.20554(0.04227)	0.09831(0.01793)	**0.04526(0.00623)**	0.39763(0.00928)	0.12834(0.08795)	0.33167(0.02930)	**0.40958(0.01436)**
UF8	0.21105(0.06693)	1.03233(0.46673)	0.26922(0.07027)	**0.19445(0.09214)**	0.16801(0.08950)	0.00000(0.00000)	0.10077(0.02094)	**0.19202(0.01859)**
UF9	0.20122(0.09586)	0.97881(0.42453)	0.31031(0.01759)	**0.18308(0.03791)**	0.31208(0.07625)	0.03669(0.05027)	0.26934(0.04147)	**0.45208(0.06781)**
UF10	1.65509(0.85125)	7.98638(1.28862)	1.39003(0.57151)	**1.36080(0.50432)**	0.00189(0.00412)	0.00019(0.00006)	0.00219(0.00058)	**0.00238(0.00084)**

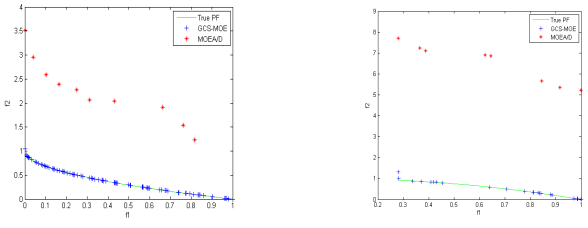
F. Additional Discussions

1) Different Criteria for the Utility function

The purpose of the utility function is to determine the trade-off between sampling in known promising regions versus sampling in under-explored regions or regions where the variance in the predictive distribution is high [50]. As discussed in Section III-D, different criteria for the utility function have been suggested for pre-screening the search space. In this section, the performances of these criteria (i.e., expected improvement (EI), Cox and John's criterion, probability of improvement (POI), and our proposed criterion) are compared through experiments. Each of these criteria is embedded in our algorithm to determine the next evaluation point. The other parameters are the same. The experimental results are shown in Table V. From these results, we can see that the proposed criterion is very effective in estimating the value of the utility function. In general, the performances of the expected improvement criterion are better than that of the probability of improvement and Cox and John's criterion. However, the algorithm with our proposed criterion performs better than the algorithm with the expected improvement criterion.

2) Effectiveness of the Proposed Surrogate Model

GCS-MOE uses a Gaussian process-based Co-subPF surrogate model (i.e., an MTGP) to predict the utility function value of the un-evaluated solutions and then searches the optimal solution using an evolutionary algorithm. To test the effect of the proposed framework, we make the comparison of GCS-MOE with standard MOEA/D on ZDT1 and ZDT6. For the parameters of MOEA/D, the population size is set to be 20, neighborhood size is set to be 5, and the number of function of evaluations is set to be 200. Fig. 4 plots the final approximations on ZDT1 and ZDT6 with the lowest IGD values among ten independent by GCS-MOE and MOEA/D. It is evident that GCS-MOE significantly outperforms MOEA/D on these test instances.



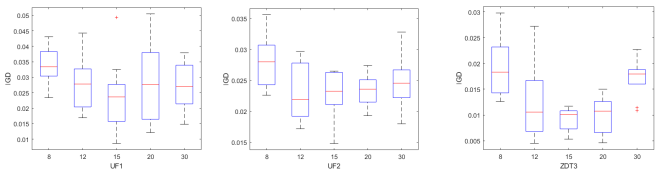
(a) ZDT1 (b) ZDT6

Fig.4 Plots of the final approximations on ZDT1 and ZDT6 obtained by GCS-MOE and MOEA/D, (a) ZDT1; (b) ZDT6.

3) Total Number of Tasks in Objective Space

In our algorithm, the subproblems are divided into M tasks to be optimized. Parameter M can take any value greater than 1. If M is too small, the obtained *PF* approximation will not provide good coverage as only a few areas of the objective space can be searched. By contrast, if M is too large, additional function evaluations may be needed to converge to

the complete *PF*. This section discusses the suitable setting of parameter M for GCS-MOE. In the experiments, M is set to 8, 12, 15, 20 and 30 for UF1, UF2 and ZDT3 instances. Detailed results are obtained by using boxplots, which represent the distributions of IGD values in the comparisons performed in the experiments. These boxplots are shown in Fig. 5. It shows that M should not be too small nor too large. Thus, M is set to 15 for all the two-objective problems and 23 for the three-objective instances in our algorithm.



(a) (b) (c)

Fig.5 Sensitivity tests for parameter M.

TABLE VI
MEAN AND STANDARD DEVIATION VALUES OF IGD AND HV OBTAINED USING DIIFFERENT m.

	HV		IGD	
Ins.	m=3	m=1	m=3	m=1
ZDT1	**0.64829(0.00301)**	0.54839(0.07144)	**0.01325(0.00293)**	0.06437(0.04062)
ZDT2	**0.30256(0.00166)**	0.26630(0.03939)	**0.00792(0.00063)**	0.03136(0.01791)
ZDT3	**0.76302(0.00295)**	0.68058(0.05946)	**0.01206(0.00126)**	0.13352(0.06561)
ZDT4	0.00000(0.00000)	0.00000(0.00000)	**1233.95204(388.51472)**	1247.27885(1193.18749)
ZDT6	**0.18027(0.04438)**	0.07536(0.06219)	**0.02365(0.01358)**	0.05511(0.01522)
UF1	**0.60223(0.01739)**	0.50908(0.06190)	**0.02625(0.00755)**	0.08185(0.03235)
UF2	**0.61985(0.00725)**	0.52670(0.04279)	**0.02538(0.00519)**	0.07645(0.02246)
UF3	**0.19335(0.16524)**	0.00068(0.00197)	**0.30028(0.15409)**	0.34761(0.00115)
UF4	**0.25862(0.00296)**	0.22610(0.01210)	**0.04238(0.00155)**	0.06240(0.00794)
UF5	**0.19880(0.06729)**	0.04467(0.07724)	**0.24214(0.07070)**	0.82302(0.57758)
UF6	**0.38582(0.02250)**	0.24707(0.05599)	**0.01859(0.01058)**	0.09112(0.02913)
UF7	**0.40958(0.01436)**	0.34258(0.05033)	**0.04526(0.00623)**	0.09185(0.05452)
UF8	**0.19202(0.01859)**	0.11114(0.02551)	**0.19445(0.09214)**	0.26980(0.07133)
UF9	**0.45208(0.06781)**	0.37909(0.07576)	**0.18308(0.03791)**	0.24379(0.06368)
UF10	**0.00238(0.00084)**	0.00207(0.00059)	**1.36080(0.50432)**	2.36366(0.56799)

4) Number of Tasks in One Multi-task GP

The number of tasks in one multi-task GP, i.e., parameter m, is important in our model. When $m = 1$, the model is changed to a common single-task GP model. To test the effect of the multi-task GP model, we compare the results of the algorithm when $m = 1$ and $m = 3$ in our experiments. The other parameters are the same as those used in the previous sections. For different values of parameter m, the algorithm independently runs 10 times in each test instance. The experimental results are shown in Table VI. From the results, it is clear that the performance of the multi-task GP model is better than that of the single-task GP model in the cases under consideration. This result shows that the multi-task learning process can estimate the model parameters confidently. This is because our approach infers multiple tasks jointly by exploiting the dependencies between tasks, such that beneficial knowledge can be transferred or shared across tasks to enhance the problem-solving. However, m cannot be set too large; otherwise, the complexity of the model learning would be too high. According to our experiments, m = 3 is suitable for our model.

VI. Conclusion

In this study, a Gaussian process-based co-subPF surrogate augmentation strategy for optimization of computationally expensive MOPs was proposed. The designed algorithm uses a decomposition approach to convert the problem of *PF* approximation into a number of scalar optimization problems. Then, these subproblems are divided into a number of tasks, where each task represents a portion or sector of the true PF (which we refer to as subPF). Notably, similarities are likely to exist across subPFs as they originate from the decomposition of the same MOP. Therefore, by inferring multiple subproblems jointly, using co-subPF multi-task GP surrogates, the dependencies among subproblems can be exploited to improve the optimization results. In particular, for expensive MOPs where the number of input observations available is small, multi-task learning augments the dataset with a number of different (but related) tasks, such that model parameters can be estimated confidently. At the same time, a novel criterion for the utility function used to determine the next candidate solution for evaluation, as well as a new model management strategy, were presented. The experimental results substantiated the efficacy of the proposed algorithm. In our future work, we plan to address many-objective optimization problems and high-dimensional problems by using a similar methodology.

Acknowledgment

This work is partially supported under the Data Science and Artificial Intelligence Center (DSAIR) and the School of Computer Science and Engineering at the Nanyang Technological University. This work is also supported by the National Natural Science Foundation of China under Grant Nos. 61301298, 61575126, and 61401283, and the Scientific Research and Development Foundation of Shenzhen under Grant Nos. JCYJ20170302145554126.

REFERENCES

[1] Tabatabaei M, Hakanen J, Hartikainen M, Miettinen K, and Sindhya K. A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. Structural and Multidisciplinary Optimization, 2015, 52(1): 1-25.

[2] Ji B, Yuan X, and Yuan Y. Modified NSGA-II for Solving Continuous Berth Allocation Problem: Using Multiobjective Constraint-Handling Strategy. IEEE transactions on cybernetics, 2017, 47(9): 2885-2895.

[3] Li H, Zhang Q, and Deng J. Biased. Multiobjective Optimization and Decomposition Algorithm. IEEE transactions on cybernetics, 2017, 47(1): 52-66.

[4] Gong YJ, Li JJ, Zhou Y, Li Y, Chung HSH, Shi YH, and Zhang J. Genetic learning particle swarm optimization. IEEE transactions on cybernetics, 2016, 46(10): 2277-2290.

[5] Zhang Q and Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[6] Wang J, Zhang W, and Zhang J. Cooperative differential evolution with multiple populations for multiobjective optimization. IEEE transactions on cybernetics, 2016, 46(12): 2848-2861.

[7] Jin Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation, 2011, 1(2): 61-70.

[8] Bhattacharya M. Reduced computation for evolutionary optimization in noisy environment.Genetic and Evolutionary Computation Conference, 2008, 2117-2122.

[9] Liang KH, Yao X, and Newton C. Evolutionary search of approximated n-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems, 2000, 4 (3): 172-183.

[10] Jin Y, Oh S, Jeon M. Incremental approximation of nonlinear constraint functions for evolutionary constrained optimization. IEEE Congress on Evolutionary Computation, 2010, 2966-2973.

[11] Chugh T, Sindhya K, Miettinen K, Hakanen J and Jin Y. On constraint handling in surrogate-assisted evolutionary many-objective optimization. Parallel Problem Solving from Nature (PPSN), Edinburgh, Scotland,2016.

[12] Lim D, Jin Y, Ong YS, and Sendhoff B. Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation, 2010, 14(3): 329-355.

[13] Samineh B, Wolfgang K, and Thomas B. Equality constraint handling for surrogate-assisted constrained optimization. 2016 IEEE Congress on Evolutionary Computation (CEC), 2016.

[14] Regis RG. Evolutionary Programming for High-Dimensional Constrained Expensive Black-Box Optimization Using Radial Basis Functions. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 326- 347.

[15] Ong YS, Nair PB, and Keane AJ. Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA journal, 2003, 41(4): 687-696.

[16] Gao G, Sun C, Zeng J, and Xue S. A Constraint Approximation Assisted PSO for Computationally Expensive Constrained Problems. Proceeding of the 11th World Congress on Intelligent Control and Automation, China,2014.

[17] Sun X, Gong D, and Li S. Classification and regression-based surrogate model -Assisted interactive genetic algorithm with individual fuzzy fitness. Genetic and Evolutionary Computation Conference, 2009, 907-914.

[18] Le MN, Ong YS, Menzel S, Jin Y, and Sendhoff B. Evolution by adapting surrogates. Evolutionary computation, 2013, 21(2): 313-340.

[19] Yu X, Jin Y, Tang K, and Yao X. Robust optimization over time - a new perspective on dynamic optimization problems. Congress on Evolutionary Computation, 2010: 3998-4003.

[20] Ong YS, Nair PB, and Lum KY. Max–min surrogate-assisted evolutionary algorithm for robust design. IEEE Transactions on Evolutionary Computation, 2006, 10(4): 392-404.

[21] Sun C, Jin Y, Cheng R, Ding J, and Zeng J. Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 644-660.

[22] Yang P, Tang K, and Yao X. Turning High-dimensional Optimization into Computationally Expensive Optimization. IEEE Transactions on Evolutionary Computation, 2017. (accepted)

[23] Sun C, Ding J, Zeng J and Jin Y. Fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. Memetic Computing, 2016. (accepted)

[24] Le MN, Ong YS, Menzel S, Seah CW, and Sendhoff B. Multi co-objective evolutionary optimization: Cross surrogate augmentation for computationally expensive problems. Evolutionary Computation (CEC), 2012 IEEE Congress on. IEEE, 2012: 1-8.

[25] Wang H, Jin Y, and Doherty J. Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Transactions on Cybernetics, 2017, 47(9): 2664-2677.

[26] Chugh T, Jin Y, Miettinen K, Hakanen J, andSindhya K. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Transactions on Evolutionary Computation, 2016. (accepted)

[27] Wang H, Jin Y and Jansen JO. Data-driven surrogate-assisted multi-objective evolutionary optimization of a trauma system. IEEE Transactions on Evolutionary Computation, 2016, 20(6): 939-952.

[28] Ong YS, Nair PB, Keane AJ, and Wong KW. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. Knowledge Incorporation in Evolutionary Computation, Springer Berlin Heidelberg, 2005: 307-331.

[29] Chugh T, Sindhya K, Miettinen K, Jin Y, Tomas Kratky, and Makkonen P. Surrogate-assisted evolutionary multiobjective shape optimization of an air intake ventilation system. Congress on Evolutionary Computation, 2017, 1541-1548.

[30] Nguyen S, Zhang M, and Tan KC. Surrogate-Assisted Genetic Programming With Simplified Models for Automated Design of Dispatching Rules. IEEE Transactions on Cybernetics, 2016. (accepted)

[31] Allmendinger R, Emmerich M, Hakanen J, Jin Y, and Rigoni E. Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case. Journal of Multi-Criteria Decision Analysis, 2017, 14(1/2): 5-25.

[32] Chugh T, Sindhya K, Hakanen J, and Miettinen K. Handling computationally expensive multiobjective optimization problems with evolutionary algorithms-A survey. Technical Report Series B, Scientific Computing No. B 4/2015, Department of Mathematical Information Technology, University of Jyvaskyla, 2015.

[33] Tian J, Tan Y, Sun C, Zeng J, Yu H and Jin Y. Comparisons of different kernels in Kriging-assisted evolutionary expensive optimization. IEEE Symposium Series on Computational Intelligence, Hawaii, USA, 2017.

[34] Yu H, Sun C, Zeng J, Tan Y and Jin Y. An adaptive model selection strategy for surrogate-assisted particle swarm optimization algorithm. IEEE Symposium on Computational Intelligence, Athens, Greece, 2016.

[35] Jones D, Schonlau M, andWelch W. Efficient global optimization of expensive black-box functions. Journal of Global Optimization, 1998, 13: 455-492.

[36] Bonilla EV, Chai KM, and Williams C. Multi-task Gaussian process prediction.Advances in neural information processing systems (NIPS), 2007: 153-160.

[37] Zhang Y, and Yeung DY. Multi-task warped gaussian process for personalized age estimation. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010.

[38] Stein ML. Interpolation of spatial data: some theory for kriging. Springer Science & Business Media, 2012.

[39] Sacks J, Welch WJ, Mitchell TJ, and Wynn HP. Design and analysis of computer experiments. Sta. Sci, 1989: 409-423.

[40] Zhou Z, Ong YS, Nair PB, Keane AJ, and Lum KY. Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions on Systems Man & Cybernetics Part C, 2007, 37(1): 66-76.

[41] Paenke I, Branke J, and Jin Y. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Transactions on Evolutionary Computation, 2006, 10(4):405-420.

[42] Liu B, Zhang Q, and Gielen GGE. A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Transactions on Evolutionary Computation, 2014, 18 (2):180-192.

[43] Jin Y. A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing. Fusion Found. Method. 2005, 9(1): 3-12.

[44] Buche D, Schraudolph NN, and Koumoutsakos P. Accelerating evolutionary algorithms with Gaussian process fitness function models. IEEE Transactions on Systems Man & Cybernetics Part C, 2005, 35(2): 184–194.

[45] Jin Y, Olhofer M, and Sendhoff B. A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation, 2002, 6(5): 481-494.

[46] Yang Q, Chen WN, Li Y, Chen CP, Xu XM, and Zhang J. Multimodal estimation of distribution algorithms. IEEE transactions on cybernetics, 2017, 47(3):636-650.

[47] Regis R and Shoemaker C. Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE Transactions on Evolutionary Computation, 2004, 8(5): 490-505.

[48] Zhang Q, Liu W, Tsang E, and Virginas B. Expensive multiobjective optimization by MOEA/D with Gaussian process model. IEEE Transactions on Evolutionary Computation, 2010, 14(3): 456-474.

[49] Keane AJ. Statistical improvement criteria for use in multiobjective design optimization. AIAA Journal, 2006, 44(4): 879-891.

[50] Emmerich M, Giannakoglou K, and Naujoks B. Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Transactions on Evolutionary Computation, 2006, 10(4): 421-439.

[51] Ponweiser W, Wagner T, Biermann D, and Vincze M. Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection. Parallel Problem Solving from Nature (PPSN X), Dortmund, Germany, 2008: 784-794.
[52] Knowles J. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation, 2006, 10(1): 50-66.
[53] Caruana R. Multitask Learning. Mach. Learn, 1997, 28(1):41-75.
[54] Thrun S. Is Learning the n-th Thing Any Easier Than Learning the First?. NIPS, 1996.
[55] Ke GY, Pan Y, Yin J, and Huang CQ. Optimizing Evaluation Metrics for Multitask Learning via the Alternating Direction Method of Multipliers. IEEE Transactions on Cybernetics, 2017. (accepted)
[56] Yu K, Tresp V, and Schwaighofer A. Learning Gaussian Processes from Multiple Tasks. In Proceedings of the 22nd International Conference on Machine Learning, 2005.
[57] Yu S, Yu K, Tresp V, and Kriegel HP. Collaborative Ordinal Regression. In Proceedings of the 23rd International Conference on Machine Learning, 2006.
[58] Wackernagel H. Multivariate geostatistics: an introduction with applications. Springer Science & Business Media, 2013.
[59] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B, 1999, 61(3):611-622.
[60] Quinonero-Candela J, Rasmussen CE, and Williams CKI. Approximation methods for gaussian process regression. Large-scale kernel machines, 2007: 203-224.
[61] Trivedi A, Srinivasan D, Sanyal K, and Ghosh A. A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition. IEEE Transactions on Evolutionary Computation, 2017, 21(3): 440-462.
[62] Yang Z, Fan Z, Zhang Q, and Cai X. Decomposition-Based-Sorting and Angle-Based-Selection for Evolutionary Multiobjective and Many-Objective Optimization. IEEE Transactions on Cybernetics, 2017,47(9): 2824-2837.
[63] Yu X, Chen WN, Gu T, Zhang H, Yuan H, Kwong S, and Zhang J.Set-Based Discrete Particle Swarm Optimization Based on Decomposition for Permutation-Based Multiobjective Combinatorial Optimization Problems.IEEE Transactions on Cybernetics, 2017. (accepted)
[64] Press W, Teukolsky S, Vetterling W, and Flannery B. Numerical Recipes in C: The Art of Scientific Computing. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[65] Cox DD and John S. SDO: a statistical method for global optimization. Multidisciplinary design optimization, V. Hampton,1997, 2: 315-329.
[66] Ulmer H, Streichert F, and Zell A. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. Evolutionary Computation (CEC'2003), Canberra, Australia, 2003: 692-699.
[67] Santner TJ, Williams BJ, and Notz WI. Some Criterion-based Experimental Designs. The Design and Analysis of Computer Experiments. Springer New York, 2003: 163-187.
[68] Liu HL, Gu F, and Zhang Q. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 450-455.
[69] Bhattacharjee KS, Singh HK, and Ray T. Multi-objective optimization with multiple spatially distributed surrogates, Journal of Mechanical Design, 2016: 1821-1831.
[70] Tian Y, Cheng R, Zhang X, and Jin Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. IEEE Computational Intelligence Magazine, 2017. (accepted)
[71] Zhang Q, Zhou A, Zhao S, Suganthany PN, Liu W, and Tiwariz S. Multiobjective optimization test instances for the CEC 2009 special session and competition.Technical Report CES-487, 2009.
[72] Deb K. Multiobjective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation, 1999, 7(3): 205-230.
[73] Zitzler E, Thiele L, Laumanns M, Fonseca CM, and Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation, 2003, 7: 117-132.
[74] Zitzler E, and Thiele L. Multiobjective optimization using evolutionary algorithms a comparative case study. Parallel Problem Solving from Nature-PPSN, LNCS, 1998: 292-301.
[75] Coello CA and Cruz Cort´es N. Solving Multiobjective Optimization Problems using an Artificial Immune System. Genetic Programming and Evolvable Machines, 2005, 6(2): 163-190.
[76] Durillo J, Nebro A, and Alba E. The jmetal framework formulti-objective optimization: Design and architecture. CEC2010, 2010:4138-4325.

Jianping Luo received his M.S. and Ph.D. degrees from College of Information Engineering of Shenzhen University, China, in 2004 and 2010, respectively. He is an associate professor of the College of Information Engineering, Shenzhen University, China. He is currently a research scholar at the Computational Intelligence Laboratory (CIL), School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include theories and applications of intelligent optimization algorithm and evolution computing.

Abhishek GUPTA holds PhD in Engineering Science from the University of Auckland, New Zealand. He is currently a Research Scientist in the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He has diverse research interests spanning the fields of continuum mechanics as well as computational intelligence, with recent emphasis on memetic computation.

Yew-Soon ONG received a PhD degree on Artificial Intelligence in complex design from the Computational Engineering and Design Center, University of Southampton, UK in 2003. He is aProfessor and Chair of the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interest in computational intelligence spans across memetic computing, evolutionary design, machine learning and Big data. He is the founding Technical Editor-in-Chief of Memetic Computing Journal, founding Chief Editor of the Springer book series on studies in adaptation, learning, and optimization, Associate Editor of the IEEE Transactions on Evolutionary Computation, the IEEE Transactions on Neural Networks & Learning Systems, IEEE Computational Intelligence Magazine, IEEE Transactions on Cybernetics, Soft Computing, International Journal of System Sciences and others.

Zhenkun Wang received the Ph.D. degree in Circuits and Systems from Xidian University, Xi ' an, China, in 2016. He is currently a Postdoctoral Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include multiobjective optimization, evolutionary computation, test problems construction and other machine learning techniques.