Chapter 1
Surrogate-assisted evolutionary optimization of large problems

Tinkle Chugh, Chaoli Sun, Handing Wang, and Yaochu Jin

Abstract This chapter presents some recent advances in surrogate-assisted evolutionary optimization of large problems. By large problems, we mean either the number of decision variables is large, or the number of objectives is large, or both. These problems pose challenges to evolutionary algorithms themselves, constructing surrogates and surrogate management. To address these challenges, we proposed two algorithms, one called kriging-assisted reference vector guided evolutionary algorithm (K-RVEA) for many-objective optimization, and the other called cooperative swarm optimization algorithm (SA-COSO) for high-dimensional single-objective optimization. Empirical studies demonstrate that K-RVEA works well for many-objective problems having up to ten objectives, while SA-COSA outperforms the state-of-the-art algorithms on 200-dimensional single-objective test problems.

Tinkle Chugh
University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora) FI-40014 University of Jyvaskyla, Finland
Department of Computer Science, University of Exeter, UK
e-mail: tinkle.chugh@gmail.com

Chaoli Sun
Department of Computer Science and Technology, Taiyuan University of Science and Technology, Shanxi 030024, China
e-mail: chaoli.sun.cn@gmail.com

Handing Wang
Department of Computer Science, University of Surrey, UK
e-mail: wanghanding.patch@gmail.com

Yaochu Jin
Department of Computer Science, University of Surrey, UK
e-mail: yaochu.jin@surrey.ac.uk

1.1 Introduction

Many industrial and real-world optimization problems involve more than three objectives and/or large number of decision variables. Evolutionary algorithms (EAs) have been widely used in the literature [12, 14, 15, 19] to solve single objective problems (SOPs) and multiobjective optimization problems (MOPs). However, their applicability to solve large optimization problems is limited. Further, problems with (computationally) expensive objective functions and/or constraints raise more challenges in using EAs. The main issue of EAs is that they consume too many function evaluations to reach a sub-optimal solution. To reduce the computational cost of the evaluation based on objective and constraint functions, several methods have been proposed in the field of surrogate-assisted optimization, where the expensive evaluations are replaced by cheap surrogates. For more details about surrogate-assisted optimization, see [10, 23]. This chapter highlights the major challenges in solving problems with a large number of objectives (known as many-objective problems) and a large number of decision variables (large-scale optimization problems). In addition, we present two recently proposed algorithms called Kriging-assisted reference vector guided EA (K-RVEA) and surrogate-assisted cooperative swarm optimization algorithm (SA-COSO) in the field of expensive many-objectives and large-scale optimization.

In last few years, evolutionary many-objective optimization has received much attention and numerous algorithms have been proposed. For more details, see a recent survey on many-objective optimization [28]. The major change to the traditional algorithms such as NSGA-II [17] or SPEA2 [60] is the reduction in the selection pressure as the number of objecitves increases. Measures to alleviate the reduction of selection pressure include modifying dominance selection criterion [13, 26, 27, 40], using decomposition in the objective space [7, 54], using some indicator to select solutions [4, 47, 59], or intrduction of a secondary selection criterion [16, 56]. Despite the existence of numerous algorithms, their applicability to real-world problems with a large number of objectives is limited. In large-scale optimization for both MOPs and SOPs, numerous EAs have been proposed. For more details about algorithms in large-scale optimization, see [34].

Practitioners and engineers in industry often need to solve problems with many-objectives and/or a large number of decision variables. In past few decades, many algorithms have been proposed in the filed of surrogate-assisted optimization. For more details about these algorithms, see [10, 23, 48]. A general framework of an SAEA is given in Figure 1.1.

As can be seen from the framework, an initial set of samples is generated e.g. by using design of experiments[32]. These samples are evaluated with computationally intensive simulations or expensive experiments to obtain their objective function values. The evaluated samples are then used to build surrogates for the given objective functions. An appropriate EA is then used with the surrogates to find samples for the next iteration. The selected samples after using the EA are evaluated with the simulations and combined with the previously evaluated ones for further training of surrogates. This process is continued until a termination criterion e.g. the allowed

Fig. 1.1: A framework representing the steps in an SAEA for solving expensive MOPs

maximum number of expensive function evaluations, is met. The best solutions from all the evaluated ones are used the final solutions. In the following, we elaborate on challenges based on the framework for large optimization problems.

1.2 Challenges for solving large optimization problems

Based on the framework in Figure 1.1, one needs to address several key challenges as follows.

1.2.1 Selection of surrogate technique

The first important challenge is to select an appropriate surrogate technique. So far, many existing regression models have been used in surrogate-assisted optimization, such as artificial neural networks [24], radial basis function networks (RBFNs) [43], polynomial repression [58], Kriging models [8], support vector regression [46], etc. Different techniques have different working methodologies and are suited for different kinds of problems. To enhance the robustness of surrogates for different problems, multiple surrogates rather than a single surrogate are combined as an ensemble [25, 21, 29, 48].

However, either a large number of objectives or a large number of decision variables give rise to challenges to SAEAs. For expensive MOPs, more than one surrogate is needed if each objective is approximated by an independent surrogate [49]. The search becomes easily misled due to the accumulated error from multiple surrogates. Therefore, a number of existing surrogate-assisted multi-objective evolutionary algorithms (SAMOEAs) employ a single surrogate to guide the selection instead of using multiple separate surrogates for objectives. For example, a performance indicator like hypervolume can be approximated using a Kriging model [41], and the population after the non-dominated sort [51] can be pre-selected by a surrogate based on classification [53]. Also for expensive large-scale problems, those well-known surrogates cannot guarantee an acceptable error and computational complexity at the same time. For instance, the complexity of the Kriging model dramatically increases with the increasing number of decision variables [9], whereas neural networks have been applied to expensive large-scale problems [10].

1.2.2 Size of training data

The second challenge is to use sufficient samples for training the surrogates. Generally, the minimal number of needed samples for a satisfactory surrogate exponentially increases as the number of decision variables increases. In other words, a large number of samples is required for training surrogates for large-scale problems. However, in many cases, only a small size data is available, which makes large-scale problems very challenging in SAEAs.

As far as we know, the current research of SAEAs mainly focuses on problems with up to 30 decision variables, which even cannot be viewed as large-scale problems. For medium- or small-size expensive problems, hundreds or thousands of exact function evaluations are still required [29, 48, 57]. In [31], the algorithm needs to reduce the dimension for the problems with 50 decision variables, otherwise there would be not enough training samples for building surrogates. Only most recently, SAEAs that are able to handle problems having more than 50 decision variables have been reported [43, 44].

Therefore, for large-scale problems, one needs to utilize the available samples appropriately considering the performance of the surrogates.

1.2.3 Selection of evolutionary algorithm

The next challenge is the choice of the EA. Ideally, any EA can be used to find samples for the next iteration. However in practice, different EAs perform differently and have sensitivity towards the number of objectives and decision variables. For example, cooperative coevolutionary algorithms [52] and the social learning particle swarm optimization algorithm [6] are good at solving large-scale problems. Moreover, MOEAs for many-objective optimization have different selection methodologies rather than Pareto dominance only that is commonly used for MOPs with two or three objectives [28].

Therefore, combining the elements of EA within the surrogate management is crucial to obtain solutions especially for large optimization problems.

1.2.4 Updating the surrogates

Another challenge which is the most important one in developing an SAEA is how to select samples for updating the surrogates, which is often known as model management or evolution control [22].

For SOPs, two types of samples are usually selected to update the surrogates. Obviously, the sample that is predicted as the optimum can help the surrogates improve the accuracy of the promising area [24, 23]. Also, selecting some samples from the region that the current surrogates cannot predict with a high confidence level fur-

ther explores the search space [5, 29, 48]. In Kriging-based SAEAs, different infill sampling criteria balancing those two types of samples are employed to select new samples [35], expected improvement (ExI) [18] and lower condence bound (LCB) [31] for instance.

The sample selection for MOPs is more complicated than for SOPs, because one needs to consider both convergence and diversity [2]. In the literature, several approaches have been used, such as selecting a set of uniformly distributed samples in the objective space [3, 9, 33] or a set of isolated samples in the decision space [1, 36], and using ExI [55], LCB [38] or expected hypervolume improvement [37, 42].

1.2.5 Computational cost for training the surrogates

One more challenge which is important especially but usually ignored in the literature is the training time of surrogates. For many-objective optimization problems, it costs more computational time if each objective is approximated by a single surrogate. For large-scale problems, the training time dramatically increases because of the training dataset in a large size. It has been shown that the training time of training a Kriging model for large-scale problems is unacceptable [8]. Therefore, it may happen that the training time is longer than evaluating an objective function and the whole aim of reducing the computation time is jeopardized.

1.3 Methods for large and expensive optimization problems

As mentioned, certain challenges exist for developing an SAEA especially for large optimization problems. Most of these challenges like selecting samples and using the elements of EAs are part of the surrogate management. Surrogate or model management is vital for the performance of an SAEA. It usually includes how to approximate, how and when to update the surrogates. In case of many-objective optimization, one needs to take care of both convergence and diversity especially in the objective space when selecting samples for training/updating the surrogates. In case of large scale problems, the purpose of the model used for, model selection and the method to select data for training/updating a model with expected performance are very important for finding a good optimal results of computationally expensive problems. Next, we present the details of two methods: K-RVEA, which was proposed for many-objective computationally expensive problems, and SA-COSO, which was proposed for large-scale complex problems.

1.3.1 A Kriging assisted reference vector guided evolutionary algorithm

Handling computationally expensive many-objective optimization problems has not received much attention in the evolutionary computation community. One reason in restraining the use of existing surrogate-based algorithms is the appropriate selection of samples to train and improve the accuracy of the surrogates and also the performance of the EA used. Moreover, the incorporation of elements of the EA used in managing the surrogates is essential for a better performance. The recently proposed algorithm K-RVEA starts the process of filling the gap between two fields focusing on many-objectives and computationally expensive function evaluations.

Two major building blocs in K-RVEA are the EA (RVEA [7]) and Kriging models [20]. RVEA is also a recently proposed EA for many-objective optimization. The envoronmental selection in RVEA is performed based on a criterion called angle penalized distance (APD) with the help of reference vectors. For more details about RVEA, see [7]. To alleviate the computational cost, Kriging models are used because of their ability to provide uncertainty information of the approximated values.

In K-RVEA, one of the major elements is the integration of the surrogates and the EA used. Most SAEAs manage the surrogates without incorporating the elements of the EA used. Therefore, the potential use of the EA used cannot be utilized properly. In K-RVEA, the incorporation of reference vectors, APD and uncertainty information from the Kriging models are used for managing the surrogates. In addition, an appropriate set of samples is selected to reduce the training time of the surrogates. In the algorithm, we use two sets of reference vectors, adaptive and fixed are used in K-RVEA for managing the surrogates, which direct the search towards the Pareto front. The algorithm has three phases as presented in Algorithm 1: : 1) initialization, 2) using RVEA with the surrogates and 3) updating the surrogates. In the algorithm, two archives $A1$ and $A2$ are used for storing the samples for training the surrogates and for storing all the evaluated samples, respectively.

In the initialization phase, an initial set of samples is generated with some design of experiment technique. These samples are evaluated with the expensive objective functions and the evaluated samples are added to two archives $A1$ and $A2$. The archive $A1$ is used to store solutions for training the Kriging models and $A2$ for storing all the evaluated samples. After building the Kriging models, Kriging models are used to approximate the objective function values. After a prefixed number of evaluations or generations with RVEA, samples are selected for updating the surrogates.

The next step in the algorithm is to select samples for updating the surrogates, which is vital in the surrogate management. Samples for updating the surrogates should be selected considering both convergence and diversity. To achieve that, we use the convergence criterion of RVEA i.e. APD, uncertainty information from the Kriging models and the reference vectors. As stated in [22], uncertain samples not only help in improving the accuracy of surrogates but also enhance the diversity.

Algorithm 1: K-RVEA

Input: FE^{max}, maximum number of expensive function evaluations
Output: nondominated solutions of all evaluated ones from $A2$
Initialization
1. Create an initial population P generated with some design of experiment technique
2. Initialize the number of function evaluations $FE = 0$ and two empty archives $A1 = A2 = \phi$
3. Evaluate the population P with the original expensive functions and add them to $A1$ and $A2$, update $FE = FE + |P|$
while $FE \leq FE^{max}$ **do**
    4. Train surrogates for each objective function by using individuals in $A1$
    Using the surrogates with RVEA*
    5. Run RVEA with Kriging models to find the individuals to update the surrogates
    Updating the surrogates
    6. Select individuals from the previous step using a selection strategy and denote the set by I
    7. Re-evaluate I with the original expensive functions and update $FE = FE + |I|$, update $A1 = A1 \cup I$ and $A2 = A2 \cup I$
    8. Remove extra individuals from $A1$ using management of training archive
    9. Go to step 4

Therefore, we select samples based on the needs of convergence and diversity. For instance, if a satisfactory degree of diversity has been achieved, we select samples using APD. Otherwise, the uncertainty from the Kriging models is used in selecting samples. To measure the need of diversity, the number of empty reference vectors is measured, which provides an indication of increase or decrease in diversity.

Selected samples are evaluated with original expensive functions and added to the archives $A1$ and $A2$. The size of the training data set can influence the training time. Therefore, to decrease the computation time further, some of the samples are removed from $A1$ in step 8 by using reference vectors. Evaluated samples in $A1$ are then used to update the surrogates. The algorithm is terminated after a prefixed maximum number of expensive function evaluations. For full details about the selection strategy and the management of the training archive, see [9].

1.3.2 A Surrogate-assisted Cooperative Swarm Optimization Algorithm

Despite the success of various surrogate techniques reported in the literature, most of these techniques have been verified only on low-dimensional problems, mainly because a large number of training samples are needed to build a sufficiently accurate surrogate for large-scale problems, which is often not affordable. The surrogate-assisted cooperative swarm optimization algorithm, called SA-COSO, aims to push the boundary of surrogate-assisted optimization techniques for solving large-scale time-consuming problems up to a dimension of 200.

Multi-swarm algorithms have shown to be effective in striking a good balance between exploration and exploitation [39]. Based on these findings, two cooperative PSO variants, one being a PSO with a constriction factor [11], called PSO in this chapter for simplicity, and the other a social learning based PSO (SL-PSO) [6], are integrated to solve computationally expensive large-scale problems. These two PSO variants cooperate in such a way that a particle in the PSO learns not only from its personal and global best particles, but also from the global best of the SL-PSO, where the particles in the SL-PSO may learn also from promising solutions contributed by the PSO. On the other hand, the SL-PSO aims to perform the exploratory search on the global surrogate model, while the PSO, assisted mainly by a local fitness estimation strategy (FES), focuses on the fast local search. The general framework of the surrogate-assisted cooperative swarm optimization algorithm is given in Figure 1.2.

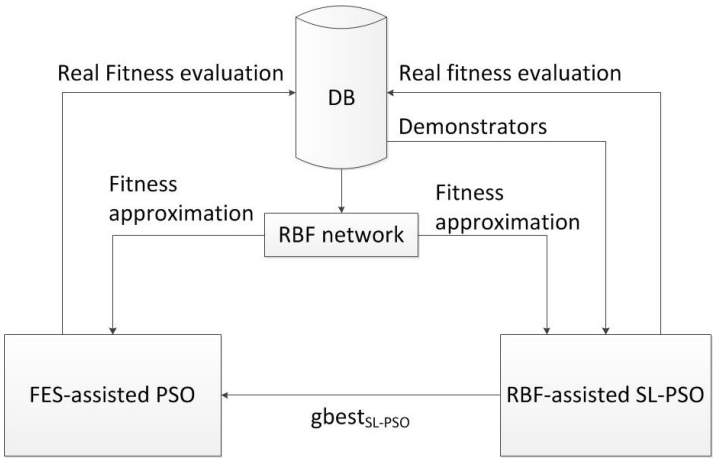


Fig. 1.2: Coupling between FES-assisted PSO and RBF-assisted SL-PSO in SA-COSO.

In Figure 1.2, *DB* represents an archive for storing the positions (decision variables) and their corresponding fitness values evaluated using the computationally expensive exact objective function. All data stored in *DB* will be utilized to train the RBFN as the surrogate model for all individuals in both FES-assisted PSO and RBF-assisted SL-PSO. Different from FES for particle swarm optimization (FESPSO) presented in [45], the fitness of a particle in the FES-assisted PSO will be approximated using the RBFN instead of the fitness evaluation using the exact objective function when the condition for using FES is not satisfied in order to further reduce the expensive fitness evaluations. Although the RBFN surrogate is also involved, the fitness of most particles in the PSO is estimated using FES. For simplicity, we term the PSO assisted mainly by the FES and sometimes also by the RBFN surrogate *FES-assisted PSO*. The particle in FES-assisted PSO will be evaluated using the

exact computationally expensive function only when its approximated fitness is potentially promising, i.e., if it is better than the current personal best, or the estimated fitness has a large degree of uncertainty. $gbest_{SL-PSO}$ in Figure 1.2 is the global best position found by the RBF-assisted SL-PSO. Since the SL-PSO algorithm is meant for global search on an RBFN global surrogate, each individual in the FES-assisted PSO will update its velocity according to Eq. (1.1) in order to avoid premature convergence into a local optimum, in which the individual learns not only from its own personal best position and the global best position of its own population, but also from the global best position obtained by the RBF-assisted SL-PSO.

$$\begin{aligned} v_{id}(t+1) = \chi(v_{id}(t) &+ c_1 r_1 (p_{id}(t) - x_{id}(t)) \\ &+ c_2 r_2 (p_{gd}(t) - x_{id}(t)) + c_3 r_3 (p_{rg,d}(t) - x_{id}(t))) \end{aligned} \tag{1.1}$$

where $1 \leq i \leq m$, r_1, r_2 and r_3 are three uniformly generated random numbers in the range $[0,1]$, c_1, c_2 and c_3 are positive constants, where c_1 is called cognitive parameter, and c_2 and c_3 are both called social coefficients. χ is the constriction factor, with

$$\chi = \frac{2k}{2 - \phi - \sqrt{(\phi^2 - 4\phi)}} \tag{1.2}$$

where ϕ is the sum of the cognitive parameter and social coefficients. In general, $\phi > 4$ and therefore, both the cognitive parameter and social coefficient are usually set to 2.05. As there are two social coefficients in Eq. (1.1), we set $c_2 + c_3$ to 2.05. k is a real number in the range $(0,1]$. $\mathbf{p}_{rg} = (p_{rg,1}, p_{rg,2}, \ldots, p_{rg,D})$ is the global best position obtained by the RBF-assisted SL-PSO ($gbest_{SL-PSO}$).

From Figure 1.2, we can see that all solutions that re-evaluated using the exact fitness function are saved in the archive *DB*. The data in the archive are used for model training on the one hand, and on the other hand, some of them will be randomly chosen to be the demonstrators in RBF-assisted SL-PSO. The position of the j-th particle will be updated as follows:

$$x_{jd}(t+1) = \begin{cases} x_{jd}(t) + \Delta x_{jd}(t+1) & \text{if } pr_j(t) \leq pr_j^L \\ x_{jd}(t) & \text{otherwise} \end{cases} \tag{1.3}$$

with

$$\Delta x_{jd}(t+1) = r_1 \cdot \Delta x_{jd}(t) + r_2 \cdot (x_{kd}(t) - x_{jd}(t)) + r_3 \cdot \varepsilon \cdot (\bar{x}_d(t) - x_{jd}(t)) \tag{1.4}$$

where $1 \leq j \leq n$, $k \in K_j$, K_j is a subset of the union of n solutions in the current SL-PSO and n solutions randomly chosen from *DB* whose fitness values are better than that of the j-th particle to be updated. pr_j, $0 \leq pr_j \leq 1$, is a randomly generated probability and pr_j^L is the probability threshold for particle j to update its position, r_1, r_2 and r_3 are three random number uniformly generated in the range $[0,1]$, x_{kd} represents the d-th ($1 \leq d \leq D$) element of particle k whose fitness is better than $f(\mathbf{x}_j)$, $\bar{x}_d(t) = \frac{\sum_{j=1}^{n} x_{jd}(t)}{n}$ is the mean position value on d-th dimension of the swarm,

ε is a parameter called the social influence factor that controls the influence of $\bar{x}_d(t)$. Please note that if no other solution is better than particle j, then the original position of this particle will be kept and participate in the evolution in the next generation.

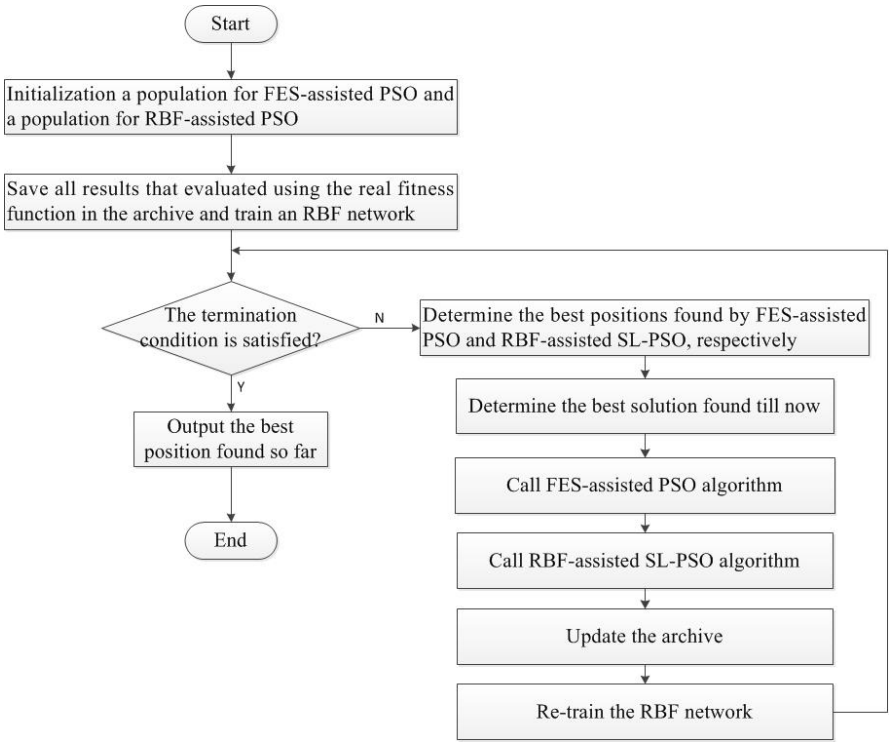


Fig. 1.3: The flowchart of SA-COSO.

Figure 1.3 gives the flowchart of the SA-COSO algorithm. Two populations will be firstly generated and evaluated using the exact fitness function for both FES-assisted PSO and RBF-assisted SL-PSO, all solutions will be then saved to an archive for RBF model training. After that, the FES-assisted PSO and RBF-assisted SL-PSO algorithms are used to find their optimal results assisted by the approximation techniques. The global best solution *gbest* of SA-COSO algorithm will be the minimum value of the best positions found by FES-assisted PSO and RBF-assisted SL-PSO, that is, $gbest = \min\{gbest_{PSO}, gbest_{SL-PSO}\}$, where $gbest_{PSO}$ and $gbest_{SL-PSO}$ are the global best positions found by FES-assisted PSO and RBF-assisted SL-PSO, respectively. In the following, we present the details of the fitness estimation strategy for PSO and the surrogate-management in SL-PSO, including training of the RBF network and update of the archive (*DB*).

1.3.2.1 FES-assisted PSO

The main steps of FES-assisted PSO are same to the canonical PSO, which includes the approaches to update the positions of individuals, to calculate the fitness values, and to determine the personal best positions and the global best position of the swarm. In the following, we will firstly give an explanation of the fitness estimation strategy used in FES-assisted PSO, and then will present a detailed description of each main step in FES-assisted PSO.

FES was firstly proposed by Sun et al. [45] based on the position relationship in PSO to approximate the fitness of the closest neighbor of particle i once its fitness is known. It has been demonstrated AS an effective approach to reduce the number of fitness evaluations for expensive optimization problems. As the PSO in SA-COSO is closely coupled with the SL-PSO, the mechanism for updating the velocity has been slightly modified as described in Eq. (1.1). Thus, the fitness estimation strategy proposed in [45] must be adapted accordingly. We rewrite the equations for updating the position of particle i and particle j ($i, j \in \{1,2,\ldots,m, i \neq j\}$), respectively as follows.

$$\begin{aligned}\mathbf{x}_i(t+1) &= \mathbf{x}_i(t) + \chi((\mathbf{x}_i(t) - \mathbf{x}_i(t-1)) + c_1\mathbf{r}_{i1}(\mathbf{p}_i(t) - \mathbf{x}_i(t)) + \\ &\quad c_2\mathbf{r}_{i2}(\mathbf{p}_g(t) - \mathbf{x}_i(t)) + c_3\mathbf{r}_{i3}(\mathbf{p}_{rg}(t) - \mathbf{x}_i(t))) \\ &= (1 + \chi(1 - c_1\mathbf{r}_{i1} - c_2\mathbf{r}_{i2} - c_3\mathbf{r}_{i3}))\mathbf{x}_i(t) - \\ &\quad \chi\mathbf{x}_i(t-1) + \chi c_1\mathbf{r}_{i1}\mathbf{p}_i(t) + \chi c_2\mathbf{r}_{i2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{i3}\mathbf{p}_{rg}(t) \end{aligned} \tag{1.5}$$

$$\begin{aligned}\mathbf{x}_j(t+1) &= \mathbf{x}_j(t) + \chi((\mathbf{x}_j(t) - \mathbf{x}_j(t-1)) + c_1\mathbf{r}_{j1}(\mathbf{p}_j(t) - \\ &\quad \mathbf{x}_j(t)) + c_2\mathbf{r}_{j2}(\mathbf{p}_g(t) - \mathbf{x}_j(t)) + c_3\mathbf{r}_{j3}(\mathbf{p}_{rg}(t) - \mathbf{x}_j(t))) \\ &= (1 + \chi(1 - c_1\mathbf{r}_{j1} - c_2\mathbf{r}_{j2} - c_3\mathbf{r}_{j3}))\mathbf{x}_j(t) - \\ &\quad \chi\mathbf{x}_j(t-1) + \chi c_1\mathbf{r}_{j1}\mathbf{p}_j(t) + \chi c_2\mathbf{r}_{j2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{j3}\mathbf{p}_{rg}(t) \end{aligned} \tag{1.6}$$

By combining and rearranging Eqs. (1.5) and (1.6), we can introduce a virtual position:

$$\begin{aligned}\mathbf{x}_v(t+1) &= \mathbf{x}_i(t+1) + \chi\mathbf{x}_i(t-1) + (1 + \chi(1 - c_1\mathbf{r}_{j1} - c_2\mathbf{r}_{j2} - c_3\mathbf{r}_{j3}))\mathbf{x}_j(t) + \\ &\quad \chi c_1\mathbf{r}_{j1}\mathbf{p}_j(t) + \chi c_2\mathbf{r}_{j2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{j3}\mathbf{p}_{rg}(t) \\ &= \mathbf{x}_j(t+1) + \chi\mathbf{x}_j(t-1) + (1 + \chi(1 - c_1\mathbf{r}_{i1} - c_2\mathbf{r}_{i2} - c_3\mathbf{r}_{i3}))\mathbf{x}_i(t) + \\ &\quad \chi c_1\mathbf{r}_{i1}\mathbf{p}_i(t) + \chi c_2\mathbf{r}_{i2}\mathbf{p}_g(t) + \chi c_3\mathbf{r}_{i3}\mathbf{p}_{rg}(t) \end{aligned} \tag{1.7}$$

Consequently, the fitness of the virtual position can be approximated using the weighted average of $f(\mathbf{x}_i(t+1))$, $f(\mathbf{x}_i(t-1))$, $f(\mathbf{x}_j(t))$, $f(\mathbf{p}_j(t))$, $f(\mathbf{p}_g(t))$ and $f(\mathbf{p}_{rg}(t))$ or of $f(\mathbf{x}_j(t+1))$, $f(\mathbf{x}_j(t-1))$, $f(\mathbf{x}_i(t))$, $f(\mathbf{p}_i(t))$, $f(\mathbf{p}_g(t))$ and $f(\mathbf{p}_{rg}(t))$ in the following form:

$$f(\mathbf{x}_v(t+1)) = \frac{WS_1}{WD_1} = \frac{WS_2}{WD_2} \tag{1.8}$$

where

$$WS_1 = \frac{f(\mathbf{x}_i(t+1))}{d_i(t+1)} + \frac{f(\mathbf{x}_i(t-1))}{d_i(t-1)} + \frac{f(\mathbf{x}_j(t))}{d_j(t)} + \frac{f(\mathbf{p}_j(t))}{d_{pj}(t)} + \frac{f(\mathbf{p}_g(t))}{d_g(t)} + \frac{f(\mathbf{p}_{rg}(t)))}{d_{rg}(t)} \tag{1.9}$$

$$WD_1 = \frac{1}{d_i(t+1)} + \frac{1}{d_i(t-1)} + \frac{1}{d_j(t)} + \frac{1}{d_{pj}(t)} + \frac{1}{d_g(t)} + \frac{1}{d_{rg}(t)} \tag{1.10}$$

$$WS_2 = \frac{f(\mathbf{x}_j(t+1))}{d_j(t+1)} + \frac{f(\mathbf{x}_j(t-1))}{d_j(t-1)} + \frac{f(\mathbf{x}_i(t))}{d_i(t)} + \frac{f(\mathbf{p}_i(t))}{d_{pi}(t)} + \frac{f(\mathbf{p}_g(t))}{d_g(t)} + \frac{f(\mathbf{p}_{rg}(t)))}{d_{rg}(t)} \tag{1.11}$$

$$WD_2 = \frac{1}{d_j(t+1)} + \frac{1}{d_j(t-1)} + \frac{1}{d_i(t)} + \frac{1}{d_{pi}(t)} + \frac{1}{d_g(t)} + \frac{1}{d_{rg}(t)} \tag{1.12}$$

where $d_i(t+1)$, $d_i(t-1)$, $d_j(t)$, $d_{pj}(t)$, $d_j(t+1)$, $d_j(t-1)$, $d_i(t)$, $d_{pi}(t)$, $d_g(t)$ and $d_{rg}(t)$ are the distances between the virtual position $\mathbf{x}_v(t+1)$ and $\mathbf{x}_i(t+1)$, $\mathbf{x}_i(t-1)$, $\mathbf{x}_j(t)$, $\mathbf{p}_j(t)$, $\mathbf{x}_j(t+1)$, $\mathbf{x}_j(t-1)$, $\mathbf{x}_i(t)$, $\mathbf{p}_i(t)$, $\mathbf{p}_g(t)$ and $\mathbf{p}_{rg}(t)$, respectively. Here, all distances are Euclidean distance.

Seen from Eqs. (1.8) to (1.12), the relationship between the fitness values of $f(\mathbf{x}_i(t+1))$ and $f(\mathbf{x}_j(t+1))$ can be established as follows:

$$\hat{f}_{FES}(\mathbf{x}_j(t+1)) = d_j(t+1) \cdot WF_{new} \tag{1.13}$$

where

$$WF_{new} = \frac{WD_1 * WS_1}{WD_2} - \frac{f(\mathbf{x}_j(t-1))}{d_j(t-1)} - \frac{f(\mathbf{x}_i(t))}{d_i(t)} - \frac{f(\mathbf{p}_i(t))}{d_{pi}(t)} - \frac{f(\mathbf{p}_g(t))}{d_g(t)} - \frac{f(\mathbf{p}_{rg}(t))}{d_{rg}(t)} \tag{1.14}$$

So Eq. (1.13) will be used to estimate the fitness of any particle j in the swarm whose fitness has not been finally determined, provided that the fitness of particle i in the same iteration is known. For each new population, all individuals will be evaluated using the real fitness function at the first generation, and otherwise the fitness of an individual will be

$$f(\mathbf{x}_j) = \begin{cases} \hat{f}_{RBF}(\mathbf{x}_j) & \text{If the fitness of the individual } j \text{ has not} \\ & \text{been approximated using Eq. (1.13)} \\ \min_{i=1,2,\ldots,j-1}\{\hat{f}^i_{FES}(\mathbf{x}_j)\} & \text{Otherwise} \end{cases} \tag{1.15}$$

where $\hat{f}_{RBF}(\mathbf{x}_j)$ is the value approximated by RBFN, and $\hat{f}^i_{FES}(\mathbf{x}_j)$ represents the value approximated by FES using the fitness value of individual i.

The personal best of all particles will then be updated in the next step. For each particle, if its fitness value in the current iteration is calculated using the RBFN and is better than its personal best, then its personal best will be replaced. However, if the current fitness value is estimated using FES and is better than the personal best, we will also compare the fitness estimated using the RBFN with the personal best. When both fitness values of the particle, one estimated using FES and the other calculated using the RBFN, are better than the personal best, we will verify the fitness of this particle using the real objective function. So eventually, the personal best of this particle will be updated only if its exact fitness value is better than the personal best.

From the above description, we can see that a particle will be evaluated using the exact fitness function only if both its fitness values estimated using the RBFN and using FES are better than its personal best. If this situation does not occur, no particle in the current iteration will be estimated using the exact fitness function, which is undesirable. To avoid false convergence, i.e., the PSO converges to a minimum of the surrogate that is not an optimum of the original fitness function, we will re-evaluate the particles if their fitness value estimated using FES has a degree of uncertainty larger than the average. The average degree of uncertainty of the estimated fitness is defined as the average difference between the fitness calculated by the RBFN and FES:

$$DF = \sum_{i=1}^{m} |f(\mathbf{x}_i) - \widehat{f}_{RBF}(\mathbf{x}_i)|/m. \tag{1.16}$$

Note that if the fitness of a particle is calculated using the RBFN, the difference will be 0. For particle i, if $|f(\mathbf{x}_i) - \widehat{f}_{RBF}(\mathbf{x}_i)|$ is larger than the mean difference DF, it will be re-evaluated using the exact fitness function and the personal best position of this particle will be updated only if the fitness value is better than the personal best.

Finally, the global best of the FES-assisted PSO needs to be updated, which is similar to the canonical PSO. The main different point is that in FES-assisted PSO, if the new global best is estimated using the RBFN or FES, it will be re-evaluated using the exact fitness function and replaces the current global best if the fitness value using the real fitness function is indeed better.

1.3.2.2 RBF-assisted SL-PSO

In the social learning particle swarm optimization algorithm, a particle (termed imitator) learns from the behaviors of different particles in the current swarm that have better fitness values (termed demonstrators) than the imitator. In this work, an RBFN is used to learn the global profile of the fitness landscape and therefore the fitness values of all particles in SL-PSO are estimated using the RBF network. Due to the fitness estimation errors introduced by the RBFN, the real fitness values of the demonstrators may be actually worse than the imitator. To avoid false convergence

of the SL-PSO, n particles stored in the *DB* will be randomly chosen as potential demonstrators for updating the particles in the current swarm. Note that all particles stored in *DB* are evaluated using the real fitness function.

Once the position and velocity of all particles are updated, their fitness value will be estimated using the RBFN and update the personal best of each particle accordingly. If the best particle (according to the RBFN) is better than the current global best, the best particle is re-evaluated using the exact fitness function. If the real fitness value of this particle is indeed better than the current global best, replace the global best with the best particle.

Note that in each iteration of the surrogate-assisted SL-PSO, at most one fitness evaluation using the real objective function will be conducted.

1.3.2.3 Update the archive *DB*

The particles stored in the archive *DB* are used not only to serve as demonstrators in SL-PSO, but also to train the RBFN, however, not all particles will be used to train the RBFN in SA-COSO in order to reduce the computational time. The size of the archive will be limited and the particles to be saved in the archive should be properly selected. As the RBFN is meant to serve as a global surrogate and the SL-PSO is supposed to perform global search, the samples for training the RBFN must be ensured that the trained RBFN can model a slightly larger region than that covered by the SL-PSO, but still be most relevant to the current swarm. For all particles that re-evaluated using the exact fitness function in the new population, the distance between their positions and those of particles in SL-PSO will be calculated. For each of them, if the distance is less than the maximum distance between the positions of particles in the archive and those of the population of SL-PSO, then this particle will be saved in the archive to replace the corresponding particle which has the largest distance in the archive to the population of SL-PSO. The authors can refer to the example presented in [44] for further understanding the strategy. Note that all distances in this work are calculated using the Euclidean distance.

1.4 Numerical experiments

In this section, we present some numerical results on standard benchmark many-objective and large scale optimization problems. First, we present the results by using the K-RVEA algorithm

1.4.1 Experimental result analysis of K-RVEA

For testing K-RVEA on many-objective optimization problems, we used DTLZ problems with three to 10 objectives. The number of decision variables is kept fixed to 10 in all problems. The algorithm is also compared with a widely used surrogate-assisted algorithm ParEGO by using hypervolume as the performance criterion. We use the same values of different parameters used in both algorithms, which are as follows:

1. Number of independent runs:15
2. Number of expensive function evaluations: 200,
3. Number of function evaluations in using EA with surrogate: 10000,
4. Crossover operator: simulated binary crossover with probability of 0.2 and distribution index of 10 and
5. Mutation operator: polynomial mutation with probability of 1/number of variables with distribution index of 50.

We used Wilcoxon statistical test with a confidence interval of 5% to compare different algorithms. The results of hypervolume with two different algorithms are shown in Table 1.1. For measuring the hypervolume, we used the worst objective function values of all runs from both the algorithms as the reference point. As can be seen, K-RVEA outperformed ParEGO in all benchmark problems used.

Nondominated solutions for a three-objective DTLZ7 problem from both the algorithms from the run with maximum hypervolume are shown in Figure 1.4. As can be seen, solutions obtained with K-RVEA are near to the Pareto front. The problem has a disconnected Pareto front. Therefore, it is difficult to find solutions in all parts of the objective space. The ParEGO algorithm randomly generates a reference vector for updating the surrogates, which does not necessary guide to the Pareto front. Therefore, for such problems, reference vector adaptation as done in K-RVEA is helpful.

A parallel coordinate plot for nondominated solutions of 10-objective DTLZ2 problem is shown in Figure 1.5. As can be seen, solutions obtained with K-RVEA has wider ranges of values than ParEGO. In other words, the K-RVEA algorithm has a better distribution of solutions in the objective space. As mentioned, the K-RVEA adaptively focuses on convergence and diversity. The DTLZ2 problem is relatively easier to solve compared to other problems in DTLZ suite. The K-RVEA algorithm major focuses on convergence for finding a representative set of Pareto optimal solutions.

1.4.2 Experimental results analysis of SA-COSO

In order to evaluate the effectiveness of SA-COSO for solving large-scale expensive optimization problems, we perform a set of experiments on 200-dimensional test problems listed in Table 1.2 and compare the performance of SA-COSO with that

Table 1.1: Comparison of K-RVEA and ParEGO using hypervolume. The results are compared with Wilcoxon statistical test of 5% confidence interval, ↑ in the table represents that K-RVEA performed better than ParEGO

Problem	k	K -RVEA				ParEGO		
		Min	Mean	Max		Min	Mean	Max
DTLZ1	3	**0.9992**	**0.9998**	**1.0000**	↑	0.9356	0.9547	0.9710
	6	**1.0000**	**1.0000**	**1.0000**	↑	0.9685	0.9768	0.9824
	8	**1.0000**	**1.0000**	**1.0000**	↑	0.9855	0.9917	0.9973
	10	**0.9996**	**0.9999**	**1.0000**	↑	0.9991	0.9995	0.9999
DTLZ2	3	**0.9212**	**0.9293**	**0.9343**	↑	0.7562	0.7867	0.8216
	6	**0.9602**	**0.9729**	**0.9846**	↑	0.7777	0.8082	0.8570
	8	**0.9512**	**0.9665**	**0.9861**	↑	0.7627	0.7946	0.8222
	10	**0.8857**	**0.9240**	**0.9558**	↑	0.6508	0.6705	0.6918
DTLZ3	3	**0.9363**	**0.9588**	**0.9837**	↑	0.8950	0.9211	0.9525
	6	**0.9665**	**0.9845**	**0.9987**	↑	0.9230	0.9461	0.9628
	8	**0.9831**	**0.9903**	**0.9992**	↑	0.9594	0.9728	0.9873
	10	**0.9985**	**0.9996**	**1.0000**	↑	0.9950	0.9971	0.9982
DTLZ4	3	**0.7557**	**0.8207**	**0.8778**	↑	0.5710	0.6368	0.7430
	6	**0.7998**	**0.8817**	**0.9334**	↑	0.5935	0.7471	0.8702
	8	**0.7782**	**0.8756**	**0.9438**	↑	0.6365	0.7333	0.8080
	10	**0.7230**	**0.8252**	**0.9147**	↑	0.4915	0.6260	0.7394
DTLZ5	3	**0.7824**	**0.7859**	**0.7914**	↑	0.6486	0.6768	0.6959
	6	**0.5848**	**0.5907**	**0.5995**	↑	0.5036	0.5214	0.5460
	8	**0.4603**	**0.4672**	**0.4746**	↑	0.3899	0.4221	0.4419
	10	**0.2217**	**0.2246**	**0.2265**	↑	0.2029	0.2109	0.2202
DTLZ6	3	**0.8928**	**0.9164**	**0.9480**	↑	0.4502	0.4592	0.4711
	6	**0.8591**	**0.9251**	**0.9520**	↑	0.4384	0.4542	0.4846
	8	**0.7443**	**0.8803**	**0.9135**	↑	0.3802	0.4039	0.4559
	10	**0.6199**	**0.6303**	**0.6399**	↑	0.2494	0.2929	0.3953
DTLZ7	3	**0.4094**	**0.4322**	**0.4528**	↑	0.2736	0.2845	0.3189
	6	**0.2771**	**0.3284**	**0.3700**	↑	0.1382	0.1678	0.2118
	8	**0.2107**	**0.3327**	**0.3705**	↑	0.0668	0.0838	0.1562
	10	**0.0688**	**0.2159**	**0.3134**	↑	0.0240	0.0438	0.1274

of PSO, FESPSO, SL-PSO, RBF-assisted SL-PSO and COSO. Refer to Table 1.3 for the definition of the algorithms investigated here. Among them, FESPSO and RBF-assisted SL-PSO are two surrogate assisted particle swarm algorithms, while the rest are not. All experimental results are obtained over 20 independent runs in Matlab®R2014b.

The sizes of pop_{PSO} and pop_{SL-PSO} in our algorithm are set to 30 and 200, respectively. In SA-SOCO and PSO, the cognitive parameters are all set to 2.05, while for the social coefficients, as it is also set to 2.05 in the canonical PSO and there are two social parameters in SA-COSO, we empirically set the social coefficient to 1.025. The parameters pr_i^L and ε in RBF-assisted SL-PSO are set to 1 and 0, respectively. As the RBFN is utilized for learning the contour of the fitness landscape, the complexity of the RBFN should not be overly large. Therefore, in the experiments, the RBFN stops learning if the maximum number of its hidden nodes (*max_node*)

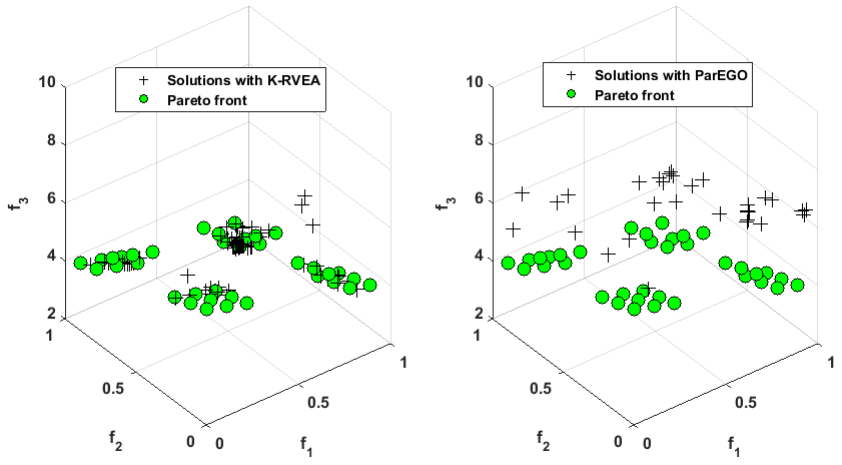


Fig. 1.4

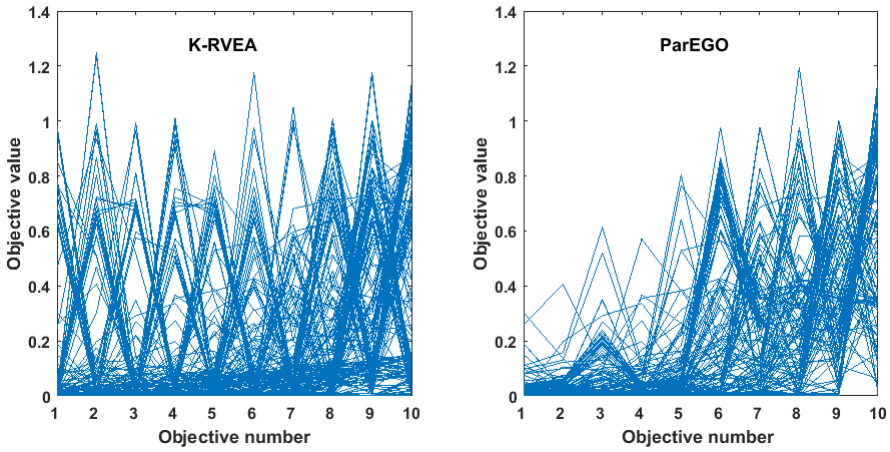


Fig. 1.5

reaches eight or the mean squared errors of the RBFN is less than 0.1. Correspondingly, the minimum size of training data can be set to $max\_node * D + max\_node$ for the RBF network with eight hidden nodes. In our experiment, the size of the archive DB (N_{DB}) is set to $max\_node * D + 10$, which is slightly larger than the minimum requirement on the size of training data. The width of the Gaussian function is set adaptively according to the number of data pairs (solutions) saved in archive DB as follows:

$$d\_max(k,d) = \max\{x_{id} | i \in \{1,2,\ldots,N_{DB}\}\} \tag{1.17}$$

Table 1.2: Characteristics of six benchmark problems.

Benchmark Problem	Description	Characteristics	Global Optimum ($f(\mathbf{x}^*)$)
F1 [30]	Ellipsoid	Unimodal	0.0
F2 [29, 30]	Rosenbrock	Multimodal with narrow valley	0.0
F3 [29, 30]	Ackley	Multimodal	0.0
F4 [29, 30]	Griewank	Multimodal	0.0
F5 [29, 30]	Shifted Rotated Rastrigin	Very complicated multimodal	-330.0
F6 [29, 30]	Rotated hybrid Composition Function	Very complicated multimodal	120.0

Table 1.3: Definition of algorithms the SA-COSO compared.

Algorithms	Definision
PSO	Particle swarm optimization with the constriction factor to update velocity [11]
FESPSO	Fitness estimation strategy assisted PSO [45]
SL-PSO	A social learning based particle swarm optimization [6]
RBF-assisted SL-PSO	SL-PSO assisted by a radial basis function network
COSO	The cooperative swarm optimization without surrogate model assistance

$$d\_min(k,d) = \min\{x_{id} | i \in \{1,2,\ldots,N_{DB}\}\} \tag{1.18}$$

$$\sigma = \frac{\sum_{k=1}^{t} \sqrt{\sum_{d=1}^{D} \|d\_max(k,d) - d\_min(k,d)\|}}{t}, \tag{1.19}$$

where $d\_max(k,d)$ and $d\_min(k,d)$ represent the maximum and minimum values of dimension d at iteration k in DB. t represents the current iteration, D is the dimension of the problem to be optimized and N_{DB} is the total number of particles (number of training samples) saved in the archive. The maximum number of fitness evaluations is set to $1,000$.

In order to make fair comparisons, the population size of both PSO and FESPSO is set to 30, the size of SL-PSO and SLPSO_RBF is set to 200, and the size of COSO is set to 230. The parameters of the RBF-assisted SL-PSO and in COSO are set the same as those in SA-COSO.

Table 1.4 lists the statistical results of all algorithms under comparison on 200-dimensional test problems averaged 20 independent runs, including the results of the Wilcoxon rank sum tests calculated at a significance level of $\alpha = 0.05$, where '$\approx$' indicates that there is no statistically significant difference between the results obtained by SA-COSO and the compared algorithm, '$+$' indicates that the compared algorithm is significantly outperformed by SA-COSO according to a Wilcoxon rank sum test, while '$-$' means that SA-COSO is significantly outperformed by the compared algorithm. From Table 1.4, we can see that SA-COSO can obtain better results on all these problems, which confirms that our method performs well on

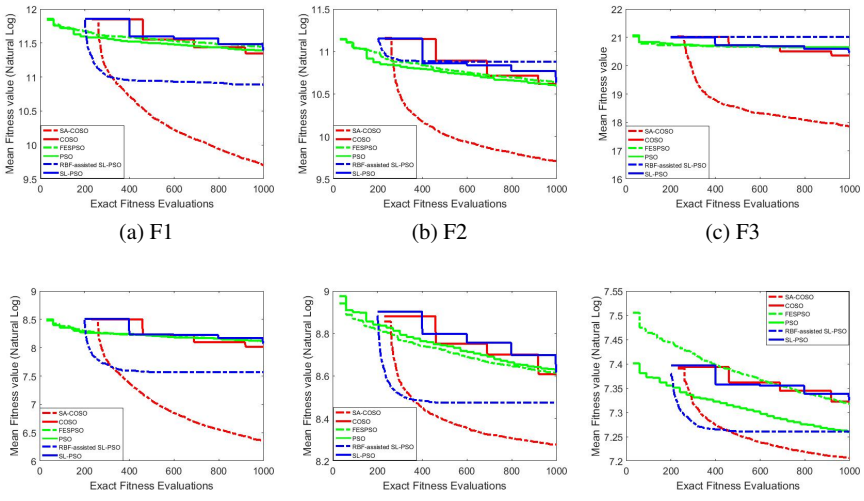
high-dimensional problems. We also plot the convergence profiles of the compared algorithms in Figure 1.6. Note that as the population sizes of the algorithms under comparison are different, the initial best fitness of the different algorithms are different. From Figure 1.6, we can see that the solution obtained by SA-COSO continuously improves as the optimization proceeds. The experimental results on 200-dimensional problems confirm the promising performance of the proposed method on the high-dimensional problems.

Table 1.4: Comparisons of the statistical results on 200-D benchmark problems.

	Approach	Mean(Wilcoxon test)	Std.		Approach	Mean(Wilcoxon test)	Std.
F1	PSO	8.7570e+04(+)	5.8963e+03	F4	PSO	3.3073e+03(+)	2.2346e+02
	FESPSO	9.2915e+04(+)	5.4782e+03		FESPSO	3.3245e+03(+)	2.8726e+02
	SL-PSO	8.3447e+04(+)	3.0600e+03		SL-PSO	2.9726e+03(+)	1.6242e+02
	RBF-assisted SL-PSO	5.3455e+04(+)	1.5658e+04		RBF-assisted SL-PSO	1.9394e+03(+)	3.5615e+02
	COSO	8.3989e+04(+)	4.5144e+03		COSO	3.0148e+03(+)	1.6510e+02
	SA-COSO	**1.6382e+04**	2.9811e+03		SA-COSO	**5.7776e+02**	1.0140e+02
F2	PSO	3.9989e+04(+)	3.0511e+03	F5	PSO	5.5872e+03(+)	3.4360e+02
	FESPSO	4.1495e+04(+)	4.4760e+03		FESPSO	5.4966e+03(+)	3.4179e+02
	SL-PSO	3.8801e+04(+)	2.4071e+03		SL-PSO	5.2454e+03(+)	1.6168e+02
	RBF-assisted SL-PSO	5.3149e+04(+)	5.5807e+03		RBF-assisted SL-PSO	4.7900e+03(+)	2.7671e+02
	COSO	3.9679e+04(+)	2.3388e+03		COSO	5.2272e+03(+)	1.5075e+02
	SA-COSO	**1.6411e+04**	4.0965e+03		SA-COSO	**3.9275e+03**	2.7254e+02
F3	PSO	2.0647e+01(+)	1.4147e-01	F6	PSO	1.4162e+03(+)	3.4170e+01
	FESPSO	2.0632e+01(+)	1.2730e-01		FESPSO	1.5035e+03(+)	3.6865e+01
	SL-PSO	2.0328e+01(+)	6.9280e-02		SL-PSO	1.5045e+03(+)	1.5512e+01
	RBF-assisted SL-PSO	2.1022e+01(+)	3.6218e-02		RBF-assisted SL-PSO	1.4228e+03(+)	2.7529e+01
	COSO	2.0356e+01(+)	1.1736e-01		COSO	1.4972e+03(+)	1.6516e+01
	SA-COSO	**1.7868e+01**	2.2319e-02		SA-COSO	**1.3473e+03**	2.4665e+01

1.5 Summary and future work

This chapter has discussed several challenges in using surrogates for large problems, i.e. problems with a large number of objectives and/or decision variables. Different algorithms are proposed for such problems which are also tested on two different real-world applications. Future work may include developing algorithms to combine the two different approaches in using surrogates for many-objectives and large-scale problems, incorporating user preferences [50], and handling constrained and mixed-integer optimization problems.



(a) F1 (b) F2 (c) F3

(d) F4 (e) F5 (f) F6

Fig. 1.6: The convergence trends for 200-dimensional F1-F6 from six algorithms.

Acknowledgements

The research of Tinkle Chugh was supported by the FiDiPro project DeCoMo funded by Tekes: Finnish Funding Agency for Innovation and Natural Environment Research Council [grant number NE/P017436/1].

References

1. Akhtar, T., Shoemaker, C.: Multi objective optimization of computationally expensive multimodal functions with RBF surrogates and multi-rule selection. Journal of Global Optimization **64**, 17–32 (2015)
2. Allmendinger, R., Emmerich, M.T.M., Hakanen, J., Jin, Y., Rigoni, E.: Surrogate-assisted multicriteria optimization: Complexities, prospective solutions, and business case. Journal of Multi-Criteria Decision Analysis **14**, 5–25 (2017)
3. Arias-Montano, A., Coello, C., Mezura-Montes, E.: Multi-objective airfoil shape optimization using a multiple-surrogate approach. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
4. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary computation **19**(1), 45–76 (2011)
5. Branke, J., Schmidt, C.: Faster convergence by means of fitness estimation. Soft Computing **9**(1), 13–20 (2005)
6. Cheng, R., Jin, Y.: A social learning particle swarm optimization algorithm for scalable optimization. Information Sciences **291**, 43–60 (2015)

7. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **20**, 773–791 (2016)
8. Chugh, T., Chakraborti, N., Sindhya, K., Jin, Y.: A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. Materials and Manufacturing Processes **32**(10), 1172–1178 (2017)
9. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Transactions on Evolutionary Computation **22**, 129 – 142 (2018)
10. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: Handling computationally expensive multiobjective optimization problems with evolutionary algorithms - A survey. Reports of the Department of Mathematical Information Technology, Series B, Scientific Computing no. B 4/2015, University of Jyvaskyla (2015)
11. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC), vol. 3, p. 1957 Vol. 3 (1999)
12. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A., et al.: Evolutionary algorithms for solving multi-objective problems, vol. 5. Springer (2007)
13. Dai, C., Wang, Y., Hu, L.: An improved {\alpha}-dominance strategy for many-objective optimization problems. Soft Computing **20**(3), 1105–1111 (2016)
14. Dasgupta, D., Michalewicz, Z.: Evolutionary algorithms in engineering applications. Springer Science & Business Media (2013)
15. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
16. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577 – 601 (2013)
17. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)
18. Emmerich, M.T., Giannakoglou, K.C., Naujoks, B.: Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation **10**(4), 421–439 (2006)
19. Fleming, P.J., Purshouse, R.C.: Evolutionary algorithms in control systems engineering: a survey. Control engineering practice **10**(11), 1223–1241 (2002)
20. Forrester, A., Keane, A.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**, 50–79 (2009)
21. Goel, T., Haftka, R.T., Shyy, W., Queipo, N.V.: Ensemble of surrogates. Structural and Multidisciplinary Optimization **33**(3), 199–216 (2007)
22. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing **9**, 3–12 (2005)
23. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61–70 (2011)
24. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on evolutionary computation **6**(5), 481–494 (2002)
25. Jin, Y., Sendhoff, B.: Reducing fitness evaluations using clustering techniques and neural network ensembles. In: Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science 3102, pp. 688–699. Springer (2004)
26. Köppen, M., Vicente-Garcia, R., Nickolay, B.: Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. In: Evolutionary multi-criterion optimization, pp. 399–412. Springer (2005)
27. Kukkonen, S., Lampinen, J.: Ranking-dominance and many-objective optimization. In: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, pp. 3983–3990. IEEE (2007)
28. Li, B., Li, J., Tang, K., Yao, X.: Many-objective evolutionary algorithms: A survey. ACM Computing Surveys (CSUR) **48**(1), 13 (2015)

29. Lim, D., Jin, Y., Ong, Y.S., Sendhoff, B.: Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation **14**(3), 329–355 (2010)
30. Liu, B., Zhang, Q., Gielen, G.: A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Transactions on Evolutionary Computation **18**(2), 180–192 (2014)
31. Liu, B., Zhang, Q., Gielen, G.G.: A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Transactions on Evolutionary Computation **18**(2), 180–192 (2014)
32. Mckay, M., Beckman, R., Conover, W.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**, 55–61 (2000)
33. Mitra, K., Majumder, S.: Successive approximate model based multi-objective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm. Chemical Engineering Science **66**, 3471–3481 (2011)
34. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Transactions on Evolutionary Computation **18**(3), 378–393 (2014)
35. Parr, J., Keane, A., Forrester, A.I., Holden, C.: Infill sampling criteria for surrogate-based optimization with constraint handling. Engineering Optimization **44**(10), 1147–1166 (2012)
36. Pilát, M., Neruda, R.: ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1202–1208. IEEE (2011)
37. Pilát, M., Neruda, R.: Hypervolume-based local search in multi-objective evolutionary optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 637–644. ACM (2014)
38. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In: G. Rudolph, T. Jansen, S. Lucas, C. Poloni, N. Beume (eds.) Proceedings of the Parallel Problem Solving from Nature-PPSN X, pp. 784–794. Springer, Berlin, Heidelberg (2008)
39. Rohler, A., Chen, S.: Multi-swarm hybrid for multi-modal optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2012)
40. Sato, H., Aguirre, H.E., Tanaka, K.: Controlling dominance area of solutions and its impact on the performance of moeas. In: International conference on evolutionary multi-criterion optimization, pp. 5–20. Springer (2007)
41. Shimoyama, K., Jeong, S., Obayashi, S.: Kriging-surrogate-based optimization considering expected hypervolume improvement in non-constrained many-objective test problems. In: Evolutionary Computation (CEC), 2013 IEEE Congress on, pp. 658–665. IEEE (2013)
42. Singh, P., Couckuyt, I., Ferranti, F., Dhaene, T.: A constrained multi-objective surrogate-based optimization algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3080–3087. IEEE (2014)
43. Sun, C., Ding, J., Zeng, J., Jin, Y.: A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. Memetic Computing pp. 1–12 (2016)
44. Sun, C., Jin, Y., Cheng, R., Ding, J., Zeng, J.: Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Transactions on Evolutionary Computation (2017)
45. Sun, C., Zeng, J., Pan, J., Xue, S., Jin, Y.: A new fitness estimation strategy for particle swarm optimization. Information Sciences **221**, 355–370 (2013)
46. Wang, D.J., Liu, F., Wang, Y.Z., Jin, Y.: A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect. Knowledge-Based Systems **90**, 70–80 (2015)
47. Wang, H., Jiao, L., Yao, X.: Two_Arch2: An improved two-archive algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **19**(4), 524–541 (2015)

48. Wang, H., Jin, Y., Doherty, J.: Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Transactions on Cybernetics **47**(9), 2664–2677 (2017)
49. Wang, H., Jin, Y., Jansen, J.: Data-driven surrogate-assisted multi-objective evolutionary optimization of a trauma system. IEEE Transactions on Evolutionary Computation **20**(6), 939–952 (2016)
50. Wang, H., Olhofer, M., Jin, Y.: Mini-review on preference modeling and articulation in multi-objective optimization: Current status and challenges. Complex & Intelligent Systems (2017). Https://link.springer.com/article/10.1007/s40747-017-0053-9
51. Wang, H., Yao, X.: Corner sort for pareto-based many-objective optimization. IEEE transactions on cybernetics **44**(1), 92–102 (2014)
52. Wiegand, R.P.: An analysis of cooperative coevolutionary algorithms. Ph.D. thesis, George Mason University Virginia (2003)
53. Zhang, J., Zhou, A., Tang, K., Zhang, G.: Preselection via classification: A case study on evolutionary multiobjective optimization. arXiv preprint arXiv:1708.01146 (2017)
54. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation **11**(6), 712–731 (2007)
55. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive multiobjective optimization by MOEA/D with Gaussian process model. IEEE Transactions on Evolutionary Computation **14**, 456–474 (2010)
56. Zhang, X., Tian, Y., Jin, Y.: A knee point driven evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **19**(6), 761–776 (2015)
57. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: 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 (2007)
58. Zhou, Z., Ong, Y.S., Nguyen, M.H., Lim, D.: A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 3, pp. 2832–2839. IEEE (2005)
59. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: International Conference on Parallel Problem Solving from Nature, pp. 832–842. Springer (2004)
60. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems pp. 95–100 (2001)