

# Evolutionary Optimization with Simplified Helper Task for High-dimensional Expensive Multiobjective Problems

XUNFENG WU, Shenzhen University

QIUZHEN LIN, Shenzhen University

JUNWEI ZHOU, Wuhan University of Technology

SONGBAI LIU, Shenzhen University

CARLOS A. COELLO COELLO, CINVESTAV-IPN

VICTOR C. M. LEUNG, Shenzhen University and The University of British Columbia

In recent years, surrogate-assisted evolutionary algorithms (SAEAs) have been sufficiently studied for tackling computationally expensive multiobjective optimization problems (EMOPs), as they can quickly estimate the qualities of solutions by using surrogate models to substitute for expensive evaluations. However, most existing SAEAs only show promising performance for solving EMOPs with no more than 10 dimensions, and become less efficient for tackling EMOPs with higher dimensionality. Thus, this article proposes a new SAEA with a simplified helper task for tackling high-dimensional EMOPs. In each generation, one simplified task will be generated artificially by using random dimension reduction on the target task (i.e., the target EMOPs). Then, two surrogate models are trained for the helper task and the target task, respectively. Based on the trained surrogate models, evolutionary multitasking optimization is run to solve these two tasks, so that the experiences of solving the helper task can be transferred to speed up the convergence of tackling the target task. Moreover, an effective model management strategy is designed to select new promising samples for training the surrogate models. When compared to five competitive SAEAs on four well-known benchmark suites, the experiments validate the advantages of the proposed algorithm on most test cases.

CCS CONCEPTS • **Computing methodologies** ~ **Artificial intelligence** ~ **Search methodologies** ~ **Continuous space search**;

**Additional Keywords and Phrases:** Surrogate-assisted evolutionary algorithm, expensive multiobjective optimization problem, evolutionary multitasking, model management

## 1 INTRODUCTION

Multiobjective optimization problems (MOPs) widely exist in many real-world applications, such as water distribution system design [Mala-Jetmarova et al. 2017], robot gripper optimization [Saravanan et al. 2009], and vehicle routing [Jozefowicz et al. 2021; Jiang et al. 2021], which have to optimize multiple (often conflicting) objectives simultaneously. Mathematically, an MOP can be defined as:

---

This work was supported by the National Natural Science Foundation of China under Grant 62173236; in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076; in part by the Shenzhen Science and Technology Program under Grant JCYJ20220531101411027; in part by the Guangdong “Pearl River Talent Recruitment Program” under Grant 2019ZT08X603; and in part by the Guangdong “Pearl River Talent Plan” under Grant 2019JC01X235. Prof. Carlos A. Coello Coello was partially supported by CONACyT grant no. 2016-01-1920.

Author’s addresses: X. Wu, Q. Lin (corresponding author), and S. Liu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail of Q. Lin: qiuzhlin@szu.edu.cn); J. Zhou is with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China. Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada. Carlos A. Coello Coello is with the Department of Computer Science, CINVESTAV-IPN (Evolutionary Computation Group), Mexico City 07300, Mexico.

$$\begin{aligned}
& \text{minimize } F(x) = (f_1(x), f_2(x), \dots, f_m(x))^T \\
& \text{subject to } x \in \prod_{i=1}^d [a_i, b_i]
\end{aligned} \tag{1}$$

where  $x = (x_1, x_2, \dots, x_d)^T$  is a solution with  $d$  decision variables,  $f_1(x), f_2(x), \dots, f_m(x)$  are  $m$  objective functions, and  $[a_i, b_i]$  for all  $i = 1, 2, \dots, d$  represent the box constraints of the search space. Due to the conflicting nature among the objectives, no single solution is able to minimize all objectives at the same time. Instead, a set of trade-off optimal solutions called the Pareto-optimal set (PS) will be found, and the mapping of PS in the objective space is called the Pareto-optimal front (PF). Multiobjective evolutionary algorithms (MOEAs) are the mainstream methods for tackling MOPs, as they can search out a set of solutions that are evenly distributed and close to the PF by evolving a population of solutions. In recent decades, a large number of MOEAs have been proposed for solving MOPs, which can be classified into three main types: Pareto-based [Zitzler et al. 2001; Deb et al. 2002], indicator-based [Zitzler and Künzli 2004; Beume et al. 2007], and decomposition-based MOEAs [Zhang and Li 2007; Liu et al. 2014; Li et al. 2014].

However, in some real-world applications, the evaluation of solutions in MOPs may need a high computational cost for its experiments [Guo et al. 2016; Chugh et al. 2017] or simulations [Ding et al. 2018; Lu et al. 2019]. These types of problems are often called expensive MOPs (EMOPs). Traditional MOEAs will encounter difficulties when tackling EMOPs, as the number of function evaluations is very limited [Jin 2011]. To address this problem, surrogate-assisted evolutionary algorithms (SAEAs) [Sun et al. 2020] have been proposed, which adopt surrogate models (also known as meta-models [Emmerich et al. 2002]) in MOEAs to substitute for expensive evaluations. Thus, even under limited computational resources, SAEAs can still obtain a good solution set to approximate the PF with the help of computationally efficient surrogate models for solving EMOPs. Many machine learning methods, such as Kriging or Gaussian processes [Chugh et al. 2018; Ath et al. 2021; Binois and Wycoff 2022], radial basis function networks (RBFNs) [Sun et al. 2017; [Lin et al. 2022], the polynomial response surface method (RSM) [Zhou et al. 2005], support vector machines (SVMs) [Kong et al. 2013; Herrera et al. 2014], random forests [Sun et al. 2020], and artificial neural networks [Jin and Sendhoff 2004], have been used as surrogate models, which are trained by truly evaluated samples. In existing SAEAs [Jin et al. 2018], surrogate models are often trained to estimate the quality of solutions by predicting their objective values [Chugh et al. 2019], their aggregated function values [Knowles 2006; Zhang et al. 2010; Tabatabaei et al. 2019], their hypervolume values [Rahat et al. 2017], or their classification [Loshchilov et al. 2010; Pan et al. 2019]. At each generation, some of the solutions will be selected and truly evaluated for further training the surrogate models to improve their prediction accuracy. As the Kriging model can provide uncertainty information, several model management strategies have been proposed, including probability of improvement (PoI) [Emmerich et al. 2006], lower confidence bound (LCB) [Torczon and Trosset 1998], and expected improvement (ExI) [Jones et al. 1998], which try to strike a balance between exploitation and exploration when selecting solutions for true function evaluations. Thus, the performance of these SAEAs highly depends on the selection of surrogate models to replace expensive evaluation and surrogate management strategies to select truly evaluated samples for improving the prediction accuracy. A detailed introduction of existing SAEAs is given in Section 2.1.

Although there are a number of SAEAs proposed with different surrogate models and surrogate management strategies, most of them only show promising performance for tackling EMOPs with no more than 10 dimensions (decision variables) and become less effective for tackling EMOPs with higher dimensionality. Therefore, this article suggests a new SAEA with a simplified helper task for tackling high-dimensional EMOPs, called SAEA-SHT. At each generation, one simplified helper task is artificially constructed from the target task (the target EMOPs), so that this helper task can be easily solved due to its low dimensionality, and then its search experiences are transferred to speed up the solution of the target task

through a surrogate-assisted evolutionary multitasking optimizer. Here, the main contributions of SAEA-SHT are the following:

- 1) One simplified helper task is generated artificially from the target task based on the random dimensionality reduction at each generation. Thus, these helper tasks are different at each generation but are all similar to the target task, which can be easily tackled. Then, surrogate models are trained by using the truly evaluated samples to assist in the solution of both the helper task and the target task.
- 2) A surrogate-assisted evolutionary multitasking optimizer is presented to simultaneously solve the helper task and the target task through knowledge transfer. In this way, the search experiences of the helper task are transferred to solve the target task, thus speeding up its convergence.
- 3) An effective model management strategy is proposed to identify promising samples for true expensive evaluations, which are used to train the surrogate models. In this way, the trained surrogate models gradually become accurate in estimating the objective values of EMOPs.

Four well-known benchmark suites (the DTLZ [Deb et al. 2002], WFG [Huband et al. 2006], UF [Zhang et al. 2008] and MaF [Cheng et al. 2017] test suites) are used to evaluate the performance of SAEA-SHT. The experimental results indicate that SAEA-SHT performs best on most instances of the test problems when compared to five competitive SAEAs (MOEA/D-EGO [Zhang et al. 2010], K-RVEA [Chugh et al. 2018], HeE-MOEA [Guo et al. 2019], ESF-RVEA [Lin et al. 2021], and KTA2 [Song et al. 2021]).

The remainder of this paper is organized as follows. Section 2 introduces some previous related work and our motivations. Section 3 presents the details of SAEA-SHT. The experimental results of SAEA-SHT with respect to five competitive SAEAs are given in Section 4. Finally, Section 5 concludes this article with some remarks for future work.

## 2 RELATED WORK AND MOTIVATIONS

Next, Section 2.1 first introduces several state-of-the-art SAEAs for solving EMOPs. Then, the motivations to design SAEA-SHT are presented in Section 2.2.

### 2.1 SAEAs for Solving EMOPs

As mentioned in Section 1, since SAEAs are able to significantly reduce the computational cost of the optimization process by approximating expensive objective functions, they have been the most popular choice for dealing with EMOPs. This section briefly introduces some state-of-the-art SAEAs for tackling EMOPs in terms of how to use surrogate models to replace expensive evaluations. Table 1 summarizes several representative SAEAs with the adopted surrogate models, the approximation of the surrogate models, and the dimensionality of the test EMOPs.

Most SAEAs apply surrogate models to approximate the objective function values, which is a straightforward method to replace expensive function evaluations. K-RVEA [Chugh et al. 2018] and ESF-RVEA [Lin et al. 2021] adopt the reference vector guided EA (RVEA) [Cheng et al. 2016] as the underlying optimizer and use surrogate models to approximate each objective function to reduce the computational cost. K-RVEA trains a Kriging model, while ESF-RVEA builds an ensemble surrogate model, which includes a global Kriging model trained under the entire search space and several Kriging submodels trained under different search subspaces. Moreover, to manage the surrogate models, K-RVEA exploits a set of reference vectors together with the uncertainty information about the approximate objective values given by the Kriging models. Then, solutions with the minimum angle penalized distance are selected if a satisfactory degree of diversity has already been maintained, while those with the maximum uncertainty are selected otherwise. ESF-RVEA proposes an effective model management strategy based on a set of reference vectors to identify new samples. First,

Table 1: Summary of Some State-of-the-art Surrogate-assisted Evolutionary Algorithms

Algorithms	Surrogate models	Approximation	Dimensions of test EMOPs
K-RVEA	Kriging	Objective function	10
ESF-RVEA	Ensemble surrogate	Objective function	10, 20, 30
HSEMA	Multiple surrogates (Kriging, RSM1, RSM2)	Objective function	10
He-MOEA	Heterogeneous ensemble (RBF1, RBF2, LSSVM)	Objective function	10, 20, 40, 80
KTA2	Influential point-insensitive Kriging	Objective function	10
EDN-ARMOEA	Efficient dropout neural network	Objective functions	20, 40, 60, 100
ParEGO	Kriging	Scalarization function	2, 3, 6, 8
MOEA/D-EGO	Kriging	Scalarization function	2, 6, 8
MCEA/D	Multiple local SVM-based classifiers	Scalarization function	50, 100, 150
SMS-EGO	Kriging	Hypervolume value	3, 6
CSEA	Feedforward neural network	Pareto dominance	10

solutions with the approximate objective values dominated by truly evaluated samples are removed to ensure convergence, and then the remaining ones closest to some randomly selected reference vectors with good diversity are selected to train the surrogate models. HSMEA [Habib et al. 2019] and HeE-MOEA [Guo et al. 2019] also use multiple surrogate models to approximate expensive objective functions. To enhance the generalization ability of the approximation, HSMEA trains multiple types of surrogate models (Kriging, RSM with polynomial degrees of 1 and 2 (RSM1 and RSM2)) and then uses one model with the minimum root mean-square error. Moreover, HSMEA adopts two sets of reference vectors, one originating from the ideal point and the other emerging from the nadir point, to first identify several solutions as candidates and then designs a local improvement scheme to improve these candidate solutions according to the Euclidean distances. HeE-MOEA proposes a heterogeneous ensemble surrogate model that can also estimate the uncertainty in approximating the objective values. This ensemble is constructed by using both feature selection and extraction to manipulate the inputs and different kinds of machine learning models (RBFN using the least squares method called RBFN1, RBFN using backpropagation method called RBFN2, and the least squares SVM called LSSVM). KTA2 [Song et al. 2021] presents an influential point-insensitive Kriging model to improve the prediction accuracy, which can reduce the negative impact of influential points without completely losing their useful information. Moreover, two sampling strategies are adaptively used in KTA2 to identify several new samples for running true expensive evaluations. More recently, EDN-ARMOEA [Guo et al. 2022] employs an efficient dropout neural network as a computationally scalable alternative of the Kriging model to approximate objective function values. In addition, a model management strategy for choosing new samples and training data is designed into an indicator-based MOEA with reference point adaptation (AR-MOEA) [Tian et al. 2018] to achieve a better balance between exploration and exploitation during the evolutionary search.

Some SAEAs use surrogate models to approximate the scalarization function values. ParEGO [Knowles 2006] extends the efficient global optimization (EGO) [Jones et al. 1998] algorithm that was originally designed for expensive single-objective optimization to tackle EMOPs. Using a randomly selected weight vector, a set of objective function values of a solution are converted to a scalarization function value, which is approximated with a trained Kriging model. MOEA/D-EGO [Zhang et al. 2010] also decomposes an EMOP into a set of single-objective optimization subproblems using weight vectors and then constructs a Kriging model to approximate the scalarization function values of each subproblem. More

recently, it was proposed MCEA/D [Sonoda and Nakata 2022] which employs multiple local SVM-based classifiers to build robust surrogate models for new samples of high-dimensional training inputs, where each local classifier is trained for a corresponding subproblem defined in decomposition-based MOEAs. In addition, MCEA/D enhances the prescreening capacity with a screening mechanism, which identifies the candidate solution closest to a region of a defined “good” category if none of them belong to this category.

Moreover, a few SAEAs train surrogate models for other purposes, such as the approximation of hypervolume values in SMS-EGO [Ponweiser et al. 2008] and the classification of solutions based on their objective values in CSEA [Pan et al. 2019]. SMS-EGO constructs a Kriging model to estimate the  $S$ -metric [Emmerich et al. 2005] of candidate solutions and then selects solutions with low  $S$ -metric values for running true expensive evaluations. CSEA first divides solutions evaluated by expensive objection functions into two categories and then trains a feedforward neural network as a surrogate model to learn the classification criterion, which is used to predict the dominance relationship between candidate solutions and reference solutions, i.e., the category of candidate solutions. In addition, CSEA splits the objective space into three regions of uncertainty by estimating a degree of reliability with a validation dataset and then picks up potentially well-converged offspring solutions based on the three regions of uncertainty.

## 2.2 Motivations

The abovementioned SAEAs are only suitable for dealing with EMOPs with no more than 10 decision variables. Nevertheless, some EMOPs modelled from real-world applications may have more than 10 decision variables, which should be properly addressed. Here, the dimensionality of the test EMOPs solved in most existing SAEAs (K-RVEA [Chugh et al. 2018], ESF-RVEA [Lin et al. 2021], HSMEA [Habib et al. 2019], HeE-MOEA [Guo et al. 2019], KTA2 [Song et al. 2021], EDN-ARMOEA [Guo et al. 2022], ParEGO [Knowles 2006], MOEA/D-EGO [Zhang et al. 2010], MCEA/D [Sonoda and Nakata 2022], SMS-EGO [Ponweiser et al. 2008], and CSEA [Pan et al. 2019]) is summarized in Table 1. According to Table 1, among these SAEAs, only HeE-MOEA, ESF-RVEA, EDN-ARMOEA, and MCEA/D are proposed for tackling EMOPs with more than 10 dimensions. However, when tackling EMOPs with higher dimensionality, such as 100 dimensions, these SAEAs do not perform as well, as shown by our experimental studies in Section 4. Therefore, further study of SAEAs is still required to address this challenge for solving high-dimensional EMOPs.

Evolutionary multitasking, as a new search paradigm in the realm of evolutionary computation, carries out evolutionary search concurrently on multiple search spaces or optimization problems [Gupta et al. 2015; Gupta et al. 2016; Ong and Gupta 2016]. Although the evolutionary multitasking paradigm was originally designed for tackling multiple optimization tasks simultaneously, some recent studies [Feng et al. 2021; Chen et al. 2021; Qiao et al. 2022; Yang et al. 2019] have shown that this paradigm is also effective for tackling complex single-tasks independently, which artificially constructs similar subtasks to speed up the convergence of tackling the original single-tasks. In [Feng et al. 2021], high-dimensional MOPs are embedded into multiple low-dimensional subspaces by using different random embeddings, where knowledge transfer is performed through a multifactorial evolutionary search in a multitask environment. In [Chen et al. 2021], several low-dimensional related tasks for feature selection are adaptively generated from a high-dimensional feature selection problem, where the crossover operator is run on the particle’s position as the knowledge transfer method to share useful information among different tasks. In [Qiao et al. 2022], a dynamic auxiliary task is created by gradually relaxing the constraints to help tackle difficult constrained MOPs, where the exchange of offspring solutions between the auxiliary task and the main task is regarded as a method of knowledge transfer. Recently, in [Yang et al. 2019], a coarse surrogate was trained in a subspace of the original search space when solving off-line data-driven EMOPs, which can perform a coarse-

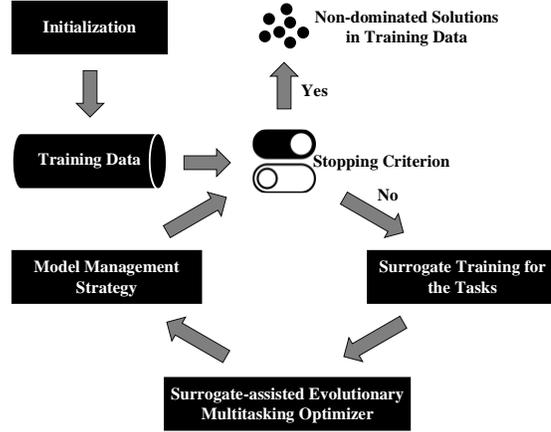


Figure 1: A flowchart of SAEA-SHT

---

### ALGORITHM 1: The Overall Framework

---

1. Initialize  $n_s$  solutions that are truly evaluated and added into the training dataset  $TD$  and set  $n_{tes} = n_s, t = 0$
  2. Set  $I = \emptyset$  and  $n_{tes} = n_s$
  3. Construct an incremental surrogate model  $tsm$  for the target task by  $TD$
  4. **while**  $n_{tes} < n_{mes}$  **do**
  5.      $(tsm, hsm, ind) \leftarrow$  **Surrogate Training for the Tasks** ( $TD, tsm, I$ ) //Algorithm 2
  6.      $P \leftarrow$  **SEMO** ( $TD, tsm, hsm, ind$ ) //Algorithm 3
  7.      $I \leftarrow$  **Model Management Strategy** ( $TD, P, I, t$ ) //Algorithm 4
  8.     Evaluate solutions in  $I$  using true objective functions and added them into  $TD$
  9.     Set  $n_{tes} = n_{tes} + |I|$  and  $t = t + 1$
  10. **end while**
  11. **return** nondominated solutions in  $TD$
- 

grained search to find the promising regions of the search space. Then, a fine-grained search is further conducted to find promising solutions by exploiting the knowledge transferred from the coarse-grained search.

Inspired by the success of using the evolutionary multitasking paradigm for solving complex single-tasks, this article also applies this paradigm and suggests a new SAEA with a simplified helper task (called SAEA-SHT) for tackling EMOPs with up to 100 dimensions. In each generation of SAEA-SHT, one simplified helper task, which can be tackled more easily, is artificially generated by randomly selecting a part of the decision variables of high-dimensional EMOPs. Then, a crossover-based transfer method is designed to exploit the optimization knowledge from the helper task to speed up the solution of high-dimensional EMOPs.

### 3 THE DETAILS OF OUR PROPOSED ALGORITHM

In this section, the details of our proposed SAEA-SHT for tackling EMOPs are introduced. In order to provide an overview of SAEA-SHT, Figure 1 illustrates the running of SAEA-SHT, which consists of four main components: 1) initialization;

2) surrogate training for the tasks; 3) surrogate-assisted evolutionary multitasking optimizer (SEMO); and 4) model management strategy. After the initialization, SAEA-SHT will iteratively run surrogate training for the tasks, SEMO, and the model management strategy until the maximum allowable number of true function evaluations is reached. Finally, the final nondominated solutions in the training data are the output of SAEA-SHT.

To clarify the running of SAEA-SHT, its pseudocode is also provided in Algorithm 1.

1) *Initialization*: The initialization process is run in line 1, which generates  $n_s$  solutions using the Latin hypercube sampling method [Mckay et al. 2000]. These solutions are stored in the training data  $\mathbf{TD}$  after evaluation by a true expensive function. Then, in line 2, the infill solution set  $\mathbf{I}$  is initialized as the empty set, the current number of true function evaluations  $n_{tes}$  is set to  $n_s$ , and the current number of generations  $t$  is set to zero. In line 3, an incremental surrogate model  $t_{sm}$  for the target task is constructed using the initial samples in  $\mathbf{TD}$ .

2) *Stopping criterion*: After the above initialization process, the evolutionary loop is run in lines 4-10 until the stopping criterion ( $n_{tes} < n_{mes}$ ) is satisfied in line 2.

3) *Surrogate training for the tasks*: At the beginning of the evolutionary loop, one simplified helper task is generated in line 5, which is a similar EMOP with some of the decision variables of the target task. Moreover, an incremental surrogate model is trained for the target task when the infill solution set  $\mathbf{I}$  is not an empty set, while another surrogate model is trained for the helper task.

4) *Surrogate-assisted evolutionary multitasking optimizer*: The surrogate-assisted evolutionary multitasking optimizer is run in line 6, in which two populations for the target task and helper task are evolved simultaneously with the help of surrogate models. Moreover, the search experiences from the helper task are utilized to assist in tackling the target task via knowledge transfer.

5) *Model management strategy*: In line 7, the population  $\mathbf{P}$  generated as an output from the surrogate-assisted evolutionary multitasking optimizer is adopted to perform the model management strategy. In this strategy, several promising new samples are chosen for updating the surrogate model for the target task.

In the following subsections, the details of the above surrogate training for the tasks, of SEMO, and of the model management strategy will be introduced, respectively.

### 3.1 Surrogate Training for the Tasks

In our method, a simplified helper task is randomly generated at each generation, which only includes some of the decision variables in high-dimensional EMOPs (the target task). Since the random dimension reduction technique is used to select some important variables and remove some useless variables, the helper tasks will be simplified and easier to solve. The helper task can be described as follows:

$$\text{minimize } F^h(x) = (f_1^h(x), f_2^h(x), \dots, f_m^h(x))^T \quad (2)$$

where  $x = (x_1, x_2, \dots, x_{n_d})^T$  is a solution with  $n_d$  decision variables partly selected from the original search space in (1), and  $f_1^h(x), f_2^h(x), \dots, f_m^h(x)$  are  $m$  simplified objective functions. Then, surrogate models are respectively trained for the target task and for the helper task. Since the Kriging model is able to provide uncertainty estimation, it is adopted as the surrogate model in our method. However, a general Kriging model needs a high training cost for dealing with high-dimensional EMOPs. Thus, an incremental Kriging model [Zhan and Xing 2021] is used to reduce the training cost for the target task, as it can gradually update the surrogate model by new coming samples instead of training from scratch in each generation. For the helper task, since its decision variables are selected randomly at each generation, the incremental Kriging model is inapplicable for the helper tasks, and thus, a general Kriging model is used for the helper task. In this way, diverse Kriging

---

**ALGORITHM 2:** Surrogate Training for the tasks ( $\mathbf{TD}$ ,  $t_{sm}$ ,  $\mathbf{I}$ )

---

1. Initialize the dimensionality of the helper task  $n_d$
  2. Randomly select  $n_d$  decision variables with their indexes saved in  $ind$
  3. Generate a helper task using the selected  $n_d$  decision variables
  4. **if**  $\mathbf{I} \neq \emptyset$  **then**
  5. Update the incremental surrogate model  $t_{sm}$  for the target task by  $\mathbf{I}$
  6. **end if**
  7.  $\mathbf{TD}_h = \mathbf{TD}(:, ind)$
  8. Construct a surrogate model  $h_{sm}$  for the helper task by  $\mathbf{TD}_h$
  9. **return**  $t_{sm}, h_{sm}, ind$
- 

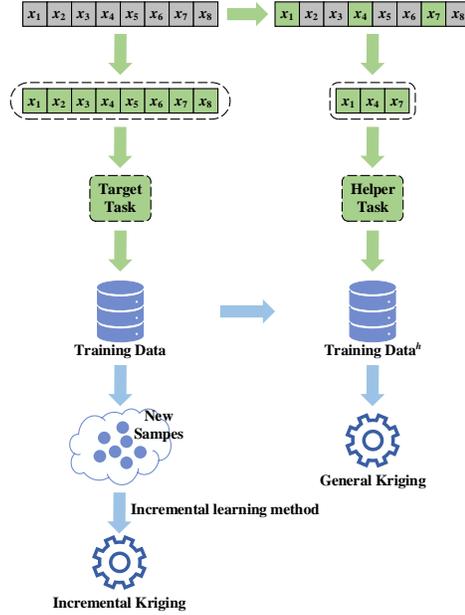


Figure 2: An example of surrogate training for the target task and helper task

models are trained for the helper tasks in the evolutionary process, which enhances the diversity of knowledge transfer from the helper tasks to the target task.

To clarify the running of the surrogate training of the target task and helper task, its pseudocode is provided in Algorithm 2 with the input training data  $\mathbf{TD}$ . In line 1 of Algorithm 2, the dimensionality of the helper task  $n_d$  is initialized. Then, in line 2,  $n_d$  decision variables are randomly selected from the original variable vectors in  $\mathbf{TD}$ , and their indexes are saved in an index array ( $ind$ ). Then, a helper task is constructed with the selected  $n_d$  decision variables in line 3. Next, in lines 4-6, for the target task, the incremental surrogate model  $t_{sm}$  is updated using the new samples in  $\mathbf{I}$  when  $\mathbf{I}$  is not an empty set. In line 7, the training data  $\mathbf{TD}_h$  for the helper task are sampled from the original training data by using the array  $ind$ . Note that if the training data for the helper task contains the same solution multiple times, only one solution from among them

is randomly selected. Afterwards, for the helper task, another surrogate model  $hsm$  is constructed using all the samples of  $\mathbf{TD}_h$  in line 8. Finally, in line 9, two trained surrogate models  $tsm$ ,  $hsm$  and the array  $ind$  are returned as the output of Algorithm 2.

Figure 2 shows an example to illustrate the surrogate training for the target task and helper task, in which the dimensions of the target task and helper task are eight and three, respectively. According to Figure 2, for the helper task, only some of the decision variables in the original problem are randomly selected, i.e.,  $x_1$ ,  $x_4$ , and  $x_7$  are selected in this case. After that, an incremental Kriging model is trained for the target task using the new samples in  $\mathbf{I}$ , while a Kriging model is trained for the helper task using the low-dimensional training data  $\mathbf{TD}_h$ . Note that although random dimensionality reduction is adopted to generate the helper task in our approach, other dimensionality reduction techniques, such as random embedding [Feng et al. 2021] and random projection [Guo et al. 2000], can also be employed.

### 3.2 Surrogate-assisted Evolutionary Multitasking Optimizer

Using the above constructed surrogate models, SEMO is further run to tackle the helper and target tasks with knowledge transfer between them. In SEMO, surrogate models are used to replace the true expensive functions for calculating the objective values of solutions.

The SEMO procedure is presented in Algorithm 3. First, in line 1, the maximum number of iterations  $g_{max}$  is initialized, and the generation counter  $g$  is set to zero. Then, a main population  $\mathbf{P}$  for the target task is created by choosing nondominated solutions from the training data  $\mathbf{TD}$  in line 2. Next, in line 3, a helper population  $\mathbf{P}_h$  with the same number of solutions as  $\mathbf{P}$  is randomly initialized for the helper task. After that, the main loop is carried out in lines 4-15, in which two populations  $\mathbf{P}$  and  $\mathbf{P}_h$  are optimized simultaneously and knowledge transfer is performed among them. In line 5, the offspring populations  $\mathbf{O}$  and  $\mathbf{O}_h$  for the main population  $\mathbf{P}$  and the helper population  $\mathbf{P}_h$ , respectively, are generated using simulated binary crossover [Deb and Agrawal 1995] and polynomial-based mutation [Deb and Goyal 1996]. In lines 6-10, another offspring population  $\mathbf{C}$  for the main population  $\mathbf{P}$  is generated through the knowledge transfer method in order to transfer the knowledge information from solutions of the helper task to the target task. Specifically, a crossover-based transfer method is used, in which the search experiences are transferred by means of a crossover operator. Different forms of crossover can be used to transfer knowledge from two solutions. In our method, simulated binary crossover is employed. To be more specific, for each solution  $s$  in the target task, in line 7, one solution  $p$  is obtained from  $s$  by only selecting decision variables in the index array  $ind$ , and a randomly selected solution  $q$  in the helper task is applied to deliver knowledge information to  $p$ . In line 8, knowledge transfer is performed between solutions  $p$  and  $q$  in the following way:

$$c(i) = 0.5 \times [(1 + \beta) \times p(i) + (1 - \beta) \times q(i)] \quad (3)$$

where  $c(i)$ ,  $p(i)$ , and  $q(i)$  represent the  $i^{\text{th}}$  decision variable of the transferred solutions  $c$ ,  $p$ , and  $q$ , respectively,  $i = 1, 2, \dots, n_d$  ( $n_d$  is the dimension of the helper task), and  $\beta$  is defined as

$$\beta = \begin{cases} (rand \times 2)^{1/\eta} & rand \leq 0.5 \\ (1/(1 - rand \times 2))^{1/\eta} & \text{otherwise} \end{cases} \quad (4)$$

where  $\eta$  is the distribution index and  $rand$  is a random number between 0 and 1. In line 9, solution  $s$  is updated by replacing the selected  $ind$  decision variables by the decision variables of the transferred solution  $c$ , and then the updated solution  $s$  is saved in  $\mathbf{C}$ . Afterwards, in line 11, two offspring populations  $\mathbf{O}$  and  $\mathbf{C}$  are evaluated using their corresponding surrogate model  $tsm$ , while the offspring population  $\mathbf{O}_h$  is evaluated using its corresponding surrogate model  $hsm$ . Then, the environmental selection of RVEA [Cheng et al. 2016] is performed in line 12 on the union population of  $\mathbf{P}$ ,  $\mathbf{O}$  and  $\mathbf{C}$  to select a new population  $\mathbf{P}$  for the target task and in line 13 on the union population of  $\mathbf{P}_h$  and  $\mathbf{O}_h$  to form a new population

---

**ALGORITHM 3: SEMO ( $TD, tsm, hsm, ind$ )**

---

1. Initialize the maximum number of iterations  $g_{max}$  and set generation counter  $g = 0, Q = \emptyset$
  2.  $P \leftarrow$  Select nondominated solutions in  $TD$  for the target task
  3.  $P_h \leftarrow$  Initialize the same number of solutions as  $P$  for the helper task
  4. **while**  $g \leq g_{max}$  **do**
  5.      $O \leftarrow$  Offspring-reproduction ( $P$ ) and  $O_h \leftarrow$  Offspring-reproduction ( $P_h$ )
  6.     **for** each solution  $s$  in  $P$  **do**
  7.          $p = s(:, ind)$  and  $q \leftarrow$  Select a solution in  $P_h$  randomly
  8.         Obtain  $c$  from  $p$  and  $q$  using (3) and (4)
  9.          $s(:, ind) = c$  and  $C = C \cup s$
  10.     **end for**
  11.     Evaluate solutions in  $O$  and  $C$  using  $tsm$ , and evaluate solutions in  $O_h$  using  $hsm$
  12.      $P \leftarrow$  Environmental-selection ( $P \cup O \cup C$ )
  13.      $P_h \leftarrow$  Environmental-selection ( $P_h \cup O_h$ )
  14.      $g = g + 1$
  15. **end while**
  16. **return**  $P$
- 

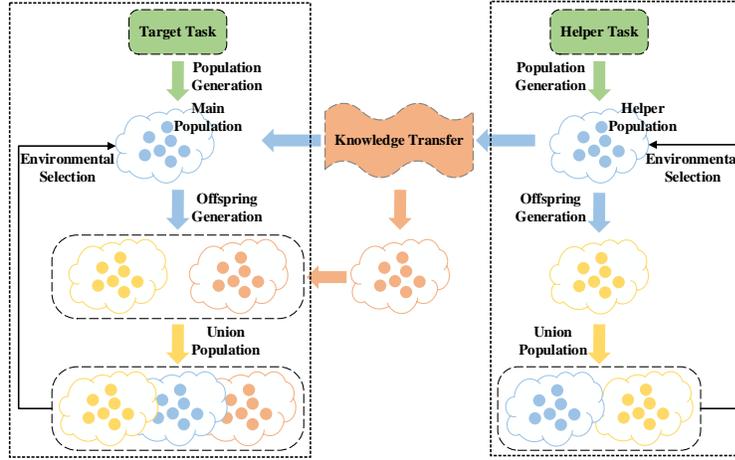


Figure 3: Process of the surrogate-assisted evolutionary multitasking optimizer

$P_h$  for the helper task. The SEMO procedures are shown in Figure 3.

### 3.3 Model Management Strategy

When SAEAs adopt a model management strategy to select solutions for updating the training data, two types of solutions (promising solutions and uncertain solutions) bring different improvements to the surrogate models [Jin et al. 2018]. Sampling promising solutions can enhance the local approximation accuracy of surrogate models in promising areas, while sampling uncertain solutions can enhance the global approximation accuracy of surrogate models in search spaces

---

**ALGORITHM 4:** Model Management Strategy ( $\mathbf{TD}, \mathbf{P}, \mathbf{I}, t$ )

---

1. Initialize the maximum number of new samples  $u$  and set  $\mathbf{C} = \emptyset, \mathbf{I} = \emptyset$
  2. **for**  $s$  in  $\mathbf{P}$  **do**
  3.     **if**  $s$  is nondominated with samples in  $\mathbf{TD}$  **then**
  4.          $\mathbf{C} \leftarrow \mathbf{C} \cup s$
  5.     **end if**
  6. **end for**
  7.  $(\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_u) \leftarrow k\text{-means}(\mathbf{C})$
  8.  $\mathbf{CM}_t \leftarrow$  Calculate the convergence metric of  $\mathbf{TD}$
  9. **if**  $\mathbf{CM}_t \leq \mathbf{CM}_{t-1}$  **then**
  10.     **for**  $i = 1$  to  $u$  **do**
  11.          $c \leftarrow$  identify the solution in  $\mathbf{C}_i$  with the minimum Euclidean distance
  12.          $\mathbf{I} = \mathbf{I} \cup c$
  13.     **end for**
  14. **else**
  15.     **for**  $i = 1$  to  $u$  **do**
  16.          $c \leftarrow$  identify the solution in  $\mathbf{C}_i$  with the maximum uncertainty
  17.          $\mathbf{I} = \mathbf{I} \cup c$
  18.     **end for**
  19. **end if**
  20. **return**  $\mathbf{I}$
- 

that have not been fully explored. Therefore, to take full advantage of the limited number of expensive function evaluations, an effective model management strategy is designed to identify infill solutions for updating training data, the details of which can be found in Algorithm 4. In line 1 of Algorithm 4, the maximum number of new samples  $u$  is initialized, an offspring population  $\mathbf{C}$  is initialized as the empty set, and an infill solution set  $\mathbf{I}$  is reset to empty. Then, in lines 2-6, solutions in  $\mathbf{P}$  that are not dominated by the samples in  $\mathbf{TD}$  are stored in  $\mathbf{C}$ . Next,  $k$ -means clustering [MacQueen 1967] is performed in  $\mathbf{C}$  to obtain  $u$  clusters  $(\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_u)$  in line 7. After that, in line 8, a convergence metric (CM) is adopted to assess the convergence of the training data  $\mathbf{TD}$ . CM is defined as follows:

$$\mathbf{CM}(\mathbf{TD}) = \frac{1}{n} \sum_{i=1}^n \|f(s_i)\| \quad (5)$$

where  $n$  is the size of  $\mathbf{TD}$  and  $s_i$  is the  $i^{\text{th}}$  solution in  $\mathbf{TD}$ . Please note that the CM values before and after the last  $\mathbf{TD}$  update are denoted by  $\mathbf{CM}_t$  and  $\mathbf{CM}_{t-1}$ , respectively. When  $\mathbf{CM}_t \leq \mathbf{CM}_{t-1}$ , which indicates that solutions in  $\mathbf{TD}$  are gradually converging; the convergence of solutions should be prioritized in this case. Thus, one solution with the minimum Euclidean distance in each cluster is identified as the new sample in line 11. Otherwise, in case that  $\mathbf{CM}_t > \mathbf{CM}_{t-1}$ , the convergence of solutions in  $\mathbf{TD}$  is not the major concern, so the priority will be moved to considering the diversity and one solution with the maximum uncertainty provided by the Kriging models in each cluster is chosen as the new sample in line 16. All the newly sampled samples are stored in  $\mathbf{I}$ . Finally,  $\mathbf{I}$  is returned as the output of Algorithm 4.

### 3.4 Computational Complexity

The computational complexity of SAEA-SHT is determined by the computational cost for function evaluations, training the surrogate models, running the surrogate-assisted evolutionary multitasking optimizer, and managing the surrogate model. More specifically, Algorithm 1 requires a time complexity of  $O(n^2)$  to construct an incremental Kriging model and  $O(n^3)$  to construct a general Kriging model in line 5. In line 6, it needs a time complexity of  $O(g_{max}n)$  to run the SEMO. In line 7, it has a time complexity of  $O(n)$  to identify the infill solutions. Therefore, the overall worst time complexity of SAEA-SHT is  $O(n^3)$  in one generation, as  $O(n^3)$  is more complex than  $O(n^2)$ ,  $O(g_{max}n)$  and  $O(n)$ .

## 4 NUMERICAL EXPERIMENTAL STUDY

### 4.1 Benchmark Problems and Performance Indicator

In our numerical experiments, the performance of the proposed SAEA-SHT is investigated for tackling the DTLZ test suite [Deb et al. 2002], the WFG test suite [Huband et al. 2006], the UF test suite [Zhang et al. 2008] and the MaF test suite [Cheng et al. 2017]. Only UF1-UF7 have two optimization objectives, while all the others have three optimization objectives. Each test problem has four sizes ( $d$ ) of decision variables ( $d = 30, 50, 70, 100$ ). The main features of the DTLZ, WFG, UF, and MaF test suites are summarized in Table A.1 of Appendix A.

The inverted generational distance (IGD) [Coello and Sierra 2004] is adopted to provide an overall assessment of all the compared algorithms. Let  $\mathbf{P}$  be a set of evenly distributed points along the PF in the objective space. Let  $\mathbf{S}$  be an approximate set obtained by the compared algorithms to the PF. Thus, IGD can be defined as the average distance from  $\mathbf{P}$  to  $\mathbf{S}$ , as follows:

$$IGD(\mathbf{S}, \mathbf{P}) = \frac{\sum_{x \in \mathbf{P}} dist(x, \mathbf{S})}{|\mathbf{P}|} \quad (6)$$

where  $dist(x, \mathbf{S})$  indicates the minimum Euclidean distance between  $x$  and the points in  $\mathbf{S}$ , and  $|\mathbf{P}|$  represents the size of  $\mathbf{P}$ . If  $\mathbf{P}$  is large enough to represent the PF very well,  $IGD(\mathbf{S}, \mathbf{P})$  is able to measure both the convergence and diversity of  $\mathbf{S}$  in a sense. To have a low  $IGD(\mathbf{S}, \mathbf{P})$ , the set  $\mathbf{S}$  must be very close to the PF and cannot miss any part of the PF.

### 4.2 Parameters Settings

In our experimental study, SAEA-SHT is compared with five competitive SAEAs (MOEA/D-EGO [Zhang et al. 2010], K-RVEA [Chugh et al. 2018], HeE-MOEA [Guo et al. 2019], ESF-RVEA [Lin et al. 2021], and KTA2 [Song et al. 2021]). The parameters of the five compared algorithms are set as suggested in their original papers, while the parameters of our proposed SAEA-SHT algorithm are summarized as follows:

- 1) Initialization: The Latin hypercube sampling method is adopted to generate  $n_s = 11 \times d - 1$  samples for the training dataset, where  $d$  is the dimensionality of the test problems.
- 2) Termination condition: The maximum number of function evaluations  $n_{mes}$  is employed as the termination condition, which is set to  $11 \times d + 120$ .
- 3) Number of independent runs: Each compared algorithm is independently run 30 times for each test problem.
- 4) Parameters settings in SAEA-SHT: The dimensionality of the helper task  $n_d$  is set to 10, the number of maximal generations  $g_{max}$  in SEMO is set to 20, and the number of infill solutions  $u$  is set to 5.

Table 2: Comparisons of SAEA-SHT with Five Algorithms on the DTLZ and WFG Test Suites

Problem	D	ESF-RVEA	MOEA/D-EGO	K-RVEA	HeE-MOEA	KTA2	SAEA-SHT
DTLZ1	30	6.2705e+2 (4.96e+1)	5.9924e+2 (1.07e+2)	5.7863e+2 (5.47e+1)	6.3110e+2 (3.76e+1)	6.1831e+2 (6.34e+1)	<b>5.4085e+2 (8.11e+1)</b>
	50	1.1714e+3 (5.21e+1)	1.1543e+3 (8.90e+1)	1.1756e+3 (6.02e+1)	1.1748e+3 (3.78e+1)	1.1686e+3 (7.24e+1)	<b>1.0306e+3 (7.90e+1)</b>
	70	1.7538e+3 (6.02e+1)	1.7205e+3 (8.41e+1)	1.7216e+3 (6.62e+1)	1.7059e+3 (6.45e+1)	1.7428e+3 (6.13e+1)	<b>1.5739e+3 (1.31e+2)</b>
	100	2.5765e+3 (7.82e+1)	2.5817e+3 (1.59e+2)	2.5846e+3 (8.09e+1)	2.4434e+3 (1.99e+2)	2.6114e+3 (6.61e+1)	<b>2.3110e+3 (1.16e+2)</b>
DTLZ2	30	1.4674e+0 (1.03e-1)	8.0864e-1 (1.09e-1)	1.1166e+0 (1.50e-1)	5.3893e-1 (7.36e-2)	1.3588e+0 (9.40e-2)	<b>4.4340e-1 (2.32e-1)</b>
	50	2.6691e+0 (1.29e-1)	2.5465e+0 (3.19e-1)	2.5410e+0 (2.10e-1)	1.2337e+0 (7.88e-2)	2.6233e+0 (2.72e-1)	<b>9.2952e-1 (2.57e-1)</b>
	70	4.0536e+0 (1.61e-1)	3.7270e+0 (5.15e-1)	3.9747e+0 (2.00e-1)	<b>1.5304e+0 (2.63e-1)</b>	3.9238e+0 (1.69e-1)	1.7304e+0 (2.64e-1)
	100	6.1108e+0 (1.41e-1)	5.9218e+0 (6.34e-1)	6.0198e+0 (2.51e-1)	<b>2.7361e+0 (1.69e-1)</b>	5.9419e+0 (1.87e-1)	3.1497e+0 (3.49e-1)
DTLZ3	30	1.7992e+3 (1.56e+2)	<b>1.3589e+3 (4.30e+2)</b>	1.7641e+3 (1.60e+2)	1.7913e+3 (9.45e+1)	1.7276e+3 (1.52e+2)	1.6083e+3 (1.92e+2)
	50	3.6433e+3 (1.94e+2)	3.5646e+3 (2.35e+2)	3.5362e+3 (2.35e+2)	3.4737e+3 (4.57e+2)	<b>3.1896e+3 (2.59e+2)</b>	3.3588e+3 (1.92e+2)
	70	5.5095e+3 (1.48e+2)	5.5281e+3 (1.43e+2)	5.4422e+3 (2.54e+2)	5.3791e+3 (3.55e+2)	5.1208e+3 (2.76e+2)	<b>5.0377e+3 (2.96e+2)</b>
	100	8.2740e+3 (2.04e+2)	7.7728e+3 (7.66e+2)	8.1407e+3 (3.17e+2)	8.1653e+3 (1.85e+2)	8.0708e+3 (4.19e+2)	<b>7.6024e+3 (4.42e+2)</b>
DTLZ4	30	1.6727e+0 (1.26e-1)	1.6734e+0 (2.16e-1)	1.6685e+0 (1.60e-1)	1.0310e+0 (4.79e-2)	<b>9.1702e-1 (2.17e-1)</b>	9.9598e-1 (3.82e-1)
	50	3.0485e+0 (1.49e-1)	3.0026e+0 (2.56e-1)	3.0668e+0 (1.36e-1)	1.4346e+0 (7.86e-2)	2.8903e+0 (2.08e-1)	<b>1.1237e+0 (2.89e-1)</b>
	70	4.3079e+0 (2.30e-1)	4.0098e+0 (8.17e-1)	4.4035e+0 (1.94e-1)	1.9926e+0 (8.03e-2)	4.1844e+0 (2.29e-1)	<b>1.8195e+0 (4.00e-1)</b>
	100	6.2813e+0 (2.23e-1)	5.9922e+0 (7.66e-1)	6.3622e+0 (2.34e-1)	<b>3.2193e+0 (2.92e-1)</b>	6.0237e+0 (2.27e-1)	3.4526e+0 (8.98e-1)
DTLZ5	30	1.3363e+0 (1.29e-1)	7.2838e-1 (1.05e-1)	1.0090e+0 (1.52e-1)	4.0426e-1 (2.84e-2)	1.2731e+0 (1.30e-1)	<b>2.8718e-1 (1.27e-1)</b>
	50	2.6320e+0 (9.86e-2)	2.5772e+0 (2.80e-1)	2.4252e+0 (4.26e-1)	9.9698e-1 (7.39e-2)	2.5056e+0 (2.08e-1)	<b>7.5162e-1 (2.20e-1)</b>
	70	3.8966e+0 (1.72e-1)	3.5791e+0 (5.71e-1)	3.8555e+0 (2.07e-1)	<b>1.4005e+0 (1.87e-1)</b>	3.8740e+0 (1.49e-1)	1.5436e+0 (3.09e-1)
	100	6.0155e+0 (2.28e-1)	5.9585e+0 (4.68e-1)	5.9360e+0 (1.76e-1)	<b>2.4331e+0 (4.85e-1)</b>	5.8165e+0 (4.79e-1)	2.7704e+0 (3.83e-1)
DTLZ6	30	1.7473e+1 (8.88e-1)	<b>1.6046e+1 (2.64e+0)</b>	2.0919e+1 (7.10e-1)	2.3710e+1 (5.41e-1)	2.2781e+1 (6.48e-1)	1.8758e+1 (1.02e+0)
	50	4.1889e+1 (2.09e-1)	<b>3.6028e+1 (3.59e+0)</b>	4.1206e+1 (8.20e-1)	4.1539e+1 (3.67e-1)	4.1395e+1 (1.03e+0)	3.6657e+1 (1.01e+0)
	70	5.9570e+1 (3.92e-1)	<b>5.3512e+1 (4.24e+0)</b>	5.9492e+1 (3.31e-1)	5.9151e+1 (5.12e-1)	5.9522e+1 (2.76e-1)	5.4904e+1 (1.06e+0)
	100	8.6411e+1 (2.86e-1)	<b>7.6307e+1 (5.92e+0)</b>	8.6044e+1 (5.21e-1)	8.5196e+1 (1.10e+0)	8.6252e+1 (4.42e-1)	8.1151e+1 (1.35e+0)
DTLZ7	30	<b>6.7371e-1 (1.12e-1)</b>	7.5217e+0 (1.32e+0)	3.9536e+0 (3.33e+0)	5.5832e+0 (7.81e-1)	8.0330e+0 (6.96e-1)	1.2860e+0 (6.87e-1)
	50	9.4700e+0 (4.35e-1)	8.8591e+0 (1.93e+0)	8.2685e+0 (2.07e+0)	7.4483e+0 (2.36e-1)	9.3107e+0 (3.94e-1)	<b>1.2331e+0 (6.12e-1)</b>
	70	9.6985e+0 (3.82e-1)	9.4573e+0 (6.02e-1)	9.9336e+0 (3.78e-1)	8.2305e+0 (5.44e-1)	9.8086e+0 (3.17e-1)	<b>2.1932e+0 (9.25e-1)</b>
	100	9.9196e+0 (2.91e-1)	9.6969e+0 (4.50e-1)	9.5235e+0 (3.20e-1)	9.2965e+0 (9.92e-1)	9.9034e+0 (3.31e-1)	<b>3.7728e+0 (4.81e-1)</b>
WFG1	30	<b>1.8148e+0 (7.91e-2)</b>	2.2111e+0 (5.21e-2)	2.0189e+0 (1.46e-1)	1.9801e+0 (5.26e-2)	1.8286e+0 (8.74e-2)	1.8363e+0 (8.59e-2)
	50	2.2513e+0 (3.20e-2)	2.1968e+0 (7.23e-2)	2.1955e+0 (1.44e-1)	1.9938e+0 (1.47e-1)	2.0989e+0 (1.17e-1)	<b>1.7708e+0 (7.15e-2)</b>
	70	2.2416e+0 (6.13e-2)	2.1881e+0 (4.88e-2)	2.2425e+0 (3.98e-2)	1.9692e+0 (9.33e-2)	2.1870e+0 (1.16e-1)	<b>1.7808e+0 (6.60e-2)</b>
	100	2.2276e+0 (5.60e-2)	2.1943e+0 (5.38e-2)	2.2299e+0 (4.96e-2)	1.9911e+0 (1.25e-1)	2.2120e+0 (7.83e-2)	<b>1.7543e+0 (7.15e-2)</b>
WFG2	30	8.1644e-1 (1.91e-2)	7.9077e-1 (5.45e-2)	7.3281e-1 (2.99e-2)	7.0746e-1 (2.48e-2)	8.4518e-1 (2.78e-2)	<b>6.0967e-1 (5.79e-2)</b>
	50	8.2628e-1 (2.35e-2)	8.2543e-1 (1.64e-2)	8.1120e-1 (3.35e-2)	7.3401e-1 (2.66e-2)	8.2561e-1 (2.34e-2)	<b>6.8105e-1 (4.12e-2)</b>
	70	8.2469e-1 (1.72e-2)	8.1193e-1 (1.08e-2)	8.1661e-1 (1.43e-2)	7.5727e-1 (1.41e-2)	8.2482e-1 (2.35e-2)	<b>7.2173e-1 (3.56e-2)</b>
	100	8.1026e-1 (1.79e-2)	8.0844e-1 (1.19e-2)	8.0436e-1 (1.15e-2)	<b>7.4055e-1 (2.07e-2)</b>	8.1033e-1 (1.17e-2)	7.6164e-1 (1.77e-2)
WFG3	30	7.5424e-1 (1.40e-2)	6.5009e-1 (2.40e-2)	7.0973e-1 (2.44e-2)	6.0957e-1 (2.48e-2)	7.5072e-1 (1.47e-2)	<b>5.0284e-1 (9.79e-2)</b>
	50	7.7242e-1 (1.19e-2)	7.7121e-1 (8.21e-3)	7.6904e-1 (1.76e-2)	6.7082e-1 (1.03e-2)	7.5771e-1 (1.03e-2)	<b>1.2331e+0 (6.12e-1)</b>
	70	7.8453e-1 (9.31e-3)	7.8401e-1 (1.02e-2)	7.8392e-1 (7.71e-3)	6.6544e-1 (2.29e-2)	7.7551e-1 (1.27e-2)	<b>6.6249e-1 (3.49e-2)</b>
	100	7.9786e-1 (5.09e-3)	7.9822e-1 (6.43e-3)	7.9849e-1 (6.51e-3)	6.8872e-1 (8.87e-3)	7.9469e-1 (1.15e-2)	<b>6.3212e-1 (4.26e-2)</b>
WFG4	30	5.6822e-1 (1.22e-2)	5.8525e-1 (4.61e-2)	5.1804e-1 (1.35e-2)	5.0608e-1 (2.67e-2)	5.9795e-1 (4.78e-2)	<b>4.7311e-1 (2.02e-2)</b>
	50	6.0020e-1 (2.02e-2)	5.7670e-1 (2.13e-2)	5.6651e-1 (2.72e-2)	4.9447e-1 (1.15e-2)	6.0435e-1 (2.31e-2)	<b>4.7816e-1 (2.14e-2)</b>
	70	5.8376e-1 (1.89e-2)	5.6570e-1 (1.31e-2)	5.7682e-1 (1.49e-2)	4.8542e-1 (1.95e-2)	5.8137e-1 (1.43e-2)	<b>4.8230e-1 (1.75e-2)</b>
	100	5.7149e-1 (1.89e-2)	5.5158e-1 (1.06e-2)	5.6903e-1 (2.09e-2)	5.0069e-1 (9.17e-3)	5.7837e-1 (2.20e-2)	<b>4.8860e-1 (1.36e-2)</b>
WFG5	30	<b>6.1600e-1 (1.89e-2)</b>	7.1551e-1 (1.59e-2)	6.6635e-1 (1.87e-2)	7.2837e-1 (1.29e-2)	7.0290e-1 (2.78e-2)	6.5469e-1 (2.54e-2)
	50	7.4762e-1 (6.26e-3)	7.2099e-1 (1.36e-2)	7.4218e-1 (1.00e-2)	7.2340e-1 (6.03e-3)	7.3044e-1 (1.29e-2)	<b>6.7682e-1 (1.62e-2)</b>
	70	7.5013e-1 (3.81e-3)	7.3466e-1 (8.24e-3)	7.4748e-1 (6.37e-3)	7.4008e-1 (6.48e-3)	7.3970e-1 (9.08e-3)	<b>7.0512e-1 (1.37e-2)</b>
	100	7.4981e-1 (3.27e-3)	7.3359e-1 (7.10e-3)	7.5004e-1 (4.89e-3)	7.4835e-1 (5.35e-3)	7.4530e-1 (3.83e-3)	<b>7.1593e-1 (9.11e-3)</b>
WFG6	30	8.8460e-1 (4.05e-2)	8.7469e-1 (2.23e-2)	8.6146e-1 (1.07e-2)	8.0717e-1 (9.30e-3)	8.7366e-1 (1.87e-2)	<b>7.9990e-1 (4.53e-2)</b>
	50	9.2031e-1 (8.96e-3)	9.2417e-1 (3.09e-2)	9.0562e-1 (1.47e-2)	<b>8.3932e-1 (1.23e-2)</b>	9.0025e-1 (1.42e-2)	8.6371e-1 (1.48e-2)
	70	9.2447e-1 (9.00e-3)	9.2098e-1 (4.57e-3)	9.2324e-1 (6.36e-3)	<b>8.4228e-1 (1.23e-2)</b>	9.1252e-1 (7.80e-3)	8.7887e-1 (2.10e-2)
	100	9.2914e-1 (4.53e-3)	9.2756e-1 (6.06e-3)	9.2664e-1 (5.21e-3)	<b>8.5282e-1 (1.50e-2)</b>	9.2369e-1 (8.69e-3)	9.0535e-1 (1.06e-2)
WFG7	30	6.8820e-1 (1.28e-2)	6.6805e-1 (1.43e-2)	6.6283e-1 (1.04e-2)	6.5873e-1 (6.71e-3)	6.9969e-1 (1.47e-2)	<b>6.5825e-1 (1.36e-2)</b>
	50	6.9738e-1 (6.33e-3)	7.0058e-1 (7.13e-3)	6.8835e-1 (8.48e-3)	6.5131e-1 (1.31e-2)	6.8690e-1 (1.17e-2)	<b>6.7769e-1 (9.01e-3)</b>
	70	6.9864e-1 (5.69e-3)	6.9881e-1 (8.08e-3)	6.9424e-1 (5.54e-3)	<b>6.5538e-1 (9.06e-3)</b>	6.9285e-1 (9.78e-3)	6.8041e-1 (5.36e-3)
	100	6.9643e-1 (5.06e-3)	6.9680e-1 (3.49e-3)	6.9454e-1 (4.92e-3)	<b>6.5843e-1 (8.15e-3)</b>	6.9305e-1 (5.15e-3)	6.8568e-1 (5.04e-3)
WFG8	30	7.7615e-1 (1.22e-2)	8.0805e-1 (2.11e-2)	<b>7.1549e-1 (1.78e-2)</b>	7.4059e-1 (2.48e-2)	8.1468e-1 (2.08e-2)	7.5029e-1 (3.28e-2)
	50	7.9517e-1 (8.52e-3)	7.9507e-1 (1.40e-2)	7.7380e-1 (2.23e-2)	<b>7.0791e-1 (1.19e-2)</b>	7.8498e-1 (1.83e-2)	7.3381e-1 (1.32e-2)
	70	7.7689e-1 (1.05e-2)	7.7778e-1 (7.89e-3)	7.6569e-1 (8.46e-3)	<b>7.0574e-1 (1.02e-2)</b>	7.6746e-1 (1.12e-2)	7.3430e-1 (1.04e-2)
	100	7.6223e-1 (1.03e-2)	7.6365e-1 (1.07e-2)	7.5772e-1 (9.72e-3)	<b>6.9467e-1 (5.68e-3)</b>	7.5775e-1 (8.11e-3)	7.2573e-1 (9.05e-3)
WFG9	30	9.2192e-1 (3.67e-2)	9.2263e-1 (4.57e-2)	8.8111e-1 (3.67e-2)	8.6834e-1 (5.65e-2)	8.5730e-1 (4.75e-2)	<b>7.6479e-1 (6.14e-2)</b>
	50	9.5073e-1 (1.62e-2)	9.5243e-1 (2.61e-2)	9.3835e-1 (2.54e-2)	9.1322e-1 (1.28e-2)	9.1210e-1 (2.83e-2)	<b>8.3944e-1 (4.06e-2)</b>
	70	9.5696e-1 (1.40e-2)	9.5944e-1 (1.35e-2)	9.4860e-1 (1.66e-2)	9.2153e-1 (6.83e-3)	9.2193e-1 (2.65e-2)	<b>8.6149e-1 (3.71e-2)</b>
	100	9.6266e-1 (1.08e-2)	9.6292e-1 (8.62e-3)	9.5801e-1 (9.21e-3)	9.4421e-1 (2.24e-2)	9.4942e-1 (1.51e-2)	<b>8.8618e-1 (3.38e-2)</b>
+/-=	3/60/1	2/58/4	1/60/3	9/37/18	1/60/3		

### 4.3 Comparisons with Five Competitive SAEAs

In this section, to investigate the optimization performance of SAEA-SHT, five competitive SAEAs (MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2) are used for performance comparison. In the following tables (Tables 2 and 3), the best result for each test instance is highlighted for easy observation. Furthermore, to obtain a statistically sound

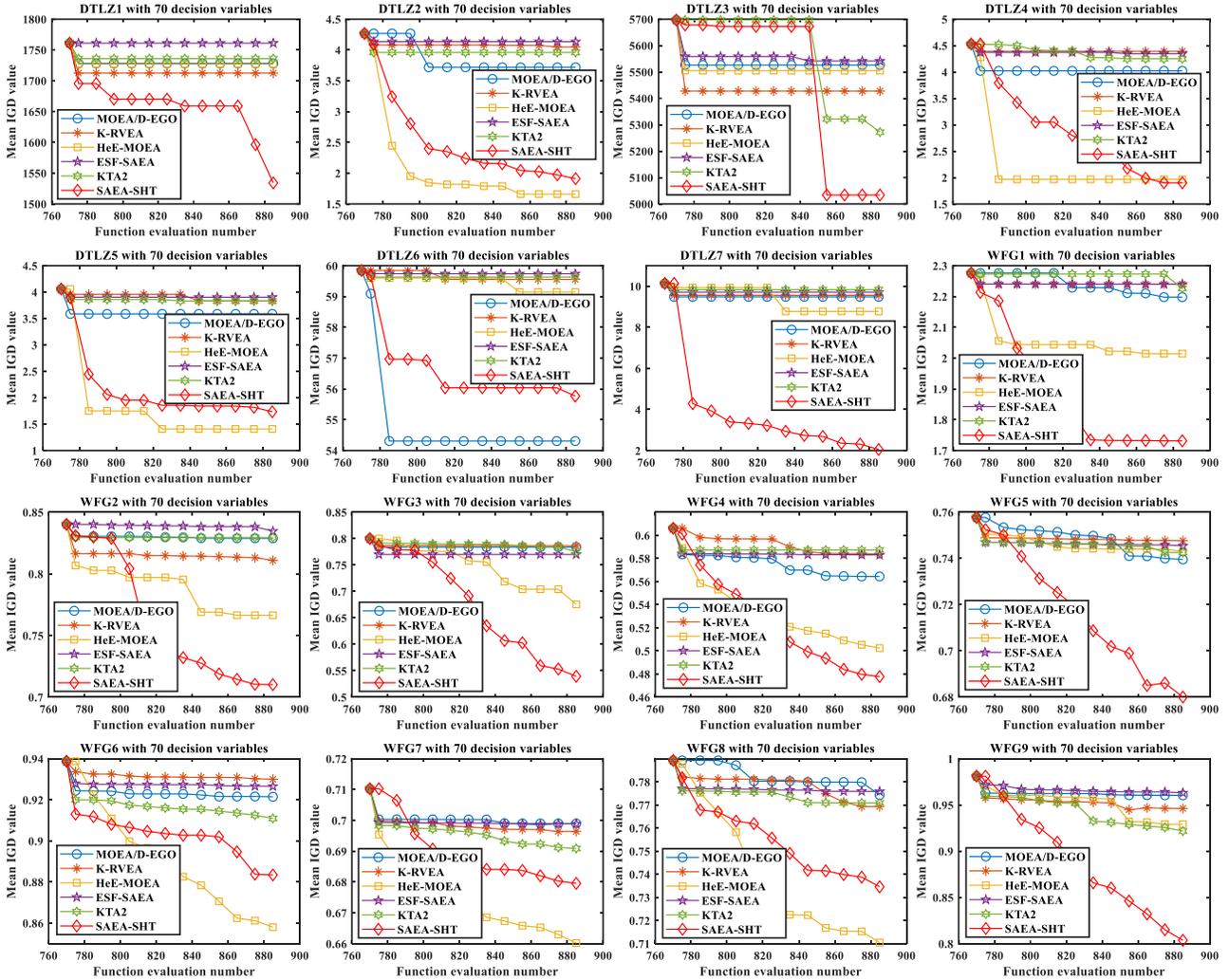


Figure 4: Convergence profiles of all the compared algorithms on the DTLZ and WFG test problems with  $d = 70$

conclusion, the Wilcoxon rank-sum test is run with a significance level of  $\alpha = 0.05$ , showing the statistically significant difference in the IGD results of SAEA-SHT and the other compared algorithms. In Tables 2 and 3, the symbols “+”, “-”, and “=” represent that the IGD results of other compared algorithms are significantly better than, worse than, and similar to those of SAEA-SHT, as indicated by the Wilcoxon rank-sum statistical test.

#### 4.3.1 Comparison Results on the DTLZ and WFG Problems

The performance of all the compared algorithms is validated on all the DTLZ and WFG test problems. The IGD mean values under 30 independent runs on the DTLZ and WFG test problems with 30, 50, 70, and 100 decision variables are provided in Table 2.

As observed from Table 2, SAEA-SHT performs best in 39 out of 64 cases, while MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2 perform best in 5, 1, 14, 3, and 2 cases, respectively. From the pairwise comparison of

SAEA-SHT with MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2, SAEA-SHT performs significantly better in 58, 60, 37, 60, and 60 cases, respectively. Specifically, for the DTLZ test problems, SAEA-SHT is outperformed by MOEA/D-EGO on 30-D and 100-D DTLZ6, by HeE-MOEA on 100-D DTLZ5, by EAF-RVEA on 30-D DTLZ7, and by KTA2 on 50-D DTLZ3, while K-RVEA fails to perform better than SAEA-SHT on any DTLZ test problem. For the WFG test problems, SAEA-SHT achieves superior performance in most WFG cases, including 50-D, 70-D, 100-D WFG1, 30-D, 50-D, 70-D WFG2, 50-D, 70-D, 100-D WFG5, 30-D WFG6, 30-D, 50-D WFG7, and in all instances of WFG3, WFG4, WFG9. MOEA/D-EGO obtains statistically better IGD results than SAEA-SHT only on 30-D WFG5. K-RVEA only achieves a significantly better IGD result on 30-D WFG8. HeE-MOEA is able to show significantly better performance on 50-D, 70-D, 100-D WFG6, 70-D, 100-D WFG7, and 50-D, 70-D, 100-D WFG8.

To further study the performance of SAEA-SHT, the convergence profiles of all the compared algorithms are depicted in Figure 4 for the DTLZ and WFG cases with 70 decision variables. It should be noted that all the IGD values in Figure 4 are calculated using all the current solutions that have been truly evaluated. From these profiles, SAEA-SHT outperforms MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2 in most cases. More specifically, SAEA-SHT is the best algorithm to achieve the smallest mean IGD results, except on DTLZ2, DTLZ5, DTLZ6, WFG6, WFG7, and WFG8. SAEA-SHT achieves significantly better performance on DTLZ1, DTLZ3, DTLZ4, DTLZ7, WFG1-WFG5, and WFG9, while it is able to show competitive performance on DTLZ2, DTLZ5, DTLZ6, WFG6, WFG7, and WFG8 according to its convergence trends in Figure 4.

In addition, the final nondominated solutions achieved by each compared algorithm are also plotted in Figure A.1 of Appendix A when solving the DTLZ and WFG problems with 70 decision variables in the run associated with the 15<sup>th</sup> best IGD value. As observed from Figure A.1 of Appendix A, only HeE-MOEA and SAEA-SHT are capable of accurately approximating the true PF of DTLZ4, while the final solutions yielded by MOEA/D-EGO, K-RVEA, ESF-RVEA, and KTA2 are far from the true PF. For DTLZ7, SAEA-SHT obtains the final solutions with better convergence, while the other compared algorithms show poor approximation performance. With respect to WFG9, which is a difficult problem to tackle, all the algorithms can only find solutions with poor convergence in the objective space. Nevertheless, SAEA-SHT still performs better in terms of convergence towards the PF.

#### 4.3.2 Comparison Results on the UF and MaF Problems

Here, the performance of all the algorithms is also compared on all the UF and MaF test problems. Their IGD values under 30 independent runs on the UF and MaF test problems are given in Table 3. As observed from Table 3, SAEA-SHT performs significantly better than MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2 in 59/64  $\approx$  92%, 52/64  $\approx$  81%, 51/64  $\approx$  80%, 46/64  $\approx$  72%, and 60/64  $\approx$  94% of the UF and MaF test problems, respectively. The smallest and largest percentages are 72% and 94%, respectively. From these IGD results, we can see that SAEA-SHT achieves the best results in 54 cases, followed by ESF-RVEA which obtains the best results in 6 cases. Therefore, it is reasonable to conclude that our SAEA-SHT algorithm performs best in most cases of the UF and MaF test problems.

For the UF test suite, our proposed algorithm SAEA-SHT is only defeated by K-RVEA in some cases, while the other compared algorithms can only achieve some similar cases with SAEA-SHT. K-RVEA performs better than SAEA-SHT on 30-D UF9 and similarly to SAEA-SHT on 30-D UF1-UF3 and UF6-UF8. When compared to SAEA-SHT, MOEA/D-EGO performs similarly on 30-D UF4, HeE-MOEA performs similarly on 100-D UF3, ESF-SAEA performs similarly on 30-D UF1- UF3, UF5-UF9, and 50-D,70-D UF5, and KTA2 performs similarly on 30-D UF3. From the IGD results of Table 3, we can also observe that our proposed algorithm SAEA-SHT shows absolute superiority over its five competitors when solving the WFG test suite. When compared with MOEA/D-EGO, except for 30-D MaF4, SAEA-SHT performs

Table 3: Comparisons of SAEA-SHT with Five Algorithms on the UF and MaF Test Suites

Problem	D	ESF-RVEA	MOEA/D-EGO	K-RVEA	HeE-MOEA	KTA2	SAEA-SHT
UF1	30	8.1323e-1 (1.08e-1) =	1.1827e+0 (1.16e-1) -	8.5522e-1 (1.85e-1) =	1.1576e+0 (8.73e-2) -	1.0429e+0 (1.63e-1) -	<b>7.5239e-1 (1.04e-1)</b>
	50	1.2156e+0 (1.52e-1) -	1.4000e+0 (8.76e-2) -	1.3496e+0 (1.24e-1) -	1.3211e+0 (1.21e-1) -	1.3350e+0 (1.08e-1) -	<b>1.0632e+0 (1.17e-1)</b>
	70	1.3671e+0 (1.33e-1) -	1.5347e+0 (7.78e-2) -	1.4967e+0 (7.98e-2) -	1.4526e+0 (7.15e-2) -	1.4840e+0 (7.61e-2) -	<b>1.1534e+0 (1.30e-1)</b>
	100	1.5125e+0 (8.25e-2) -	1.5997e+0 (5.88e-2) -	<b>1.6265e+0 (5.92e-2) -</b>	1.5430e+0 (6.08e-2) -	1.5871e+0 (6.92e-2) -	<b>1.3029e+0 (1.33e-1)</b>
UF2	30	1.7861e-1 (2.45e-2) =	4.0388e-1 (1.19e-1) -	<b>1.7177e-1 (1.31e-2) =</b>	2.3752e-1 (4.11e-2) -	4.5843e-1 (6.80e-2) -	1.7488e-1 (1.61e-2)
	50	5.6076e-1 (1.03e-1) -	6.7117e-1 (3.08e-2) -	5.9238e-1 (1.23e-1) -	4.0217e-1 (3.86e-2) -	6.4641e-1 (3.16e-2) -	<b>2.1792e-1 (1.69e-2)</b>
	70	6.3404e-1 (5.75e-2) -	7.0410e-1 (3.05e-2) -	6.9988e-1 (2.77e-2) -	4.1328e-1 (7.09e-2) -	6.9267e-1 (1.92e-2) -	<b>2.5505e-1 (3.08e-2)</b>
	100	6.6568e-1 (4.42e-2) -	7.4549e-1 (1.80e-2) -	7.3977e-1 (1.98e-2) -	4.6805e-1 (2.98e-2) -	7.3451e-1 (2.44e-2) -	<b>2.8910e-1 (2.05e-2)</b>
UF3	30	5.7617e-1 (1.52e-1) =	9.6293e-1 (1.39e-1) -	5.6243e-1 (2.88e-2) =	6.6134e-1 (2.52e-2) -	6.2540e-1 (1.75e-1) =	<b>5.4848e-1 (2.42e-2)</b>
	50	8.5095e-1 (1.57e-1) -	9.9277e-1 (6.39e-2) -	9.8126e-1 (4.95e-2) -	6.5419e-1 (1.77e-2) -	9.5446e-1 (1.45e-1) -	<b>5.2924e-1 (3.81e-2)</b>
	70	8.8879e-1 (8.54e-2) -	1.0042e+0 (3.29e-2) -	9.4552e-1 (4.36e-2) -	6.3771e-1 (4.93e-2) -	9.8653e-1 (5.31e-2) -	<b>5.5071e-1 (4.27e-2)</b>
	100	9.0911e-1 (5.90e-2) -	9.7933e-1 (3.61e-2) -	9.6608e-1 (4.75e-2) -	7.0755e-1 (4.85e-2) =	9.4588e-1 (5.23e-2) -	<b>6.0890e-1 (4.80e-2)</b>
UF4	30	1.5719e-1 (9.03e-3) -	1.4368e-1 (1.15e-2) =	1.6122e-1 (7.01e-3) -	1.7609e-1 (5.28e-3) -	1.7308e-1 (4.51e-3) -	<b>1.4198e-1 (8.38e-3)</b>
	50	1.8707e-1 (2.75e-3) -	1.8370e-1 (3.64e-3) -	1.8902e-1 (4.69e-3) -	1.8613e-1 (1.01e-3) -	1.8318e-1 (7.59e-3) -	<b>1.7111e-1 (6.38e-3)</b>
	70	1.9259e-1 (2.37e-3) -	1.9166e-1 (3.55e-3) -	1.9590e-1 (1.74e-3) -	1.9321e-1 (1.51e-3) -	1.9196e-1 (4.67e-3) -	<b>1.8202e-1 (4.82e-3)</b>
	100	1.9685e-1 (1.96e-3) -	1.9705e-1 (2.35e-3) -	1.9763e-1 (1.80e-3) -	1.9764e-1 (1.66e-3) -	1.9516e-1 (2.84e-3) -	<b>1.9259e-1 (2.44e-3)</b>
UF5	30	<b>3.7071e+0 (5.85e-1) =</b>	4.8256e+0 (2.84e-1) -	4.4210e+0 (3.75e-1) -	4.8578e+0 (1.63e-1) -	4.6999e+0 (3.05e-1) -	4.0302e+0 (4.15e-1)
	50	4.8642e+0 (4.07e-1) =	5.4749e+0 (1.08e-1) -	5.3638e+0 (1.56e-1) -	5.4219e+0 (1.06e-1) -	5.1640e+0 (3.24e-1) -	<b>4.6321e+0 (4.68e-1)</b>
	70	5.1778e+0 (1.58e-1) -	5.7517e+0 (1.87e-1) -	5.7250e+0 (1.66e-1) -	5.6173e+0 (1.90e-1) -	5.6214e+0 (2.42e-1) -	<b>4.9570e+0 (3.30e-1)</b>
	100	5.7485e+0 (2.06e-1) -	5.8768e+0 (2.16e-1) -	6.0618e+0 (1.55e-1) -	5.7953e+0 (1.22e-1) -	5.9692e+0 (1.52e-1) -	<b>5.3280e+0 (3.03e-1)</b>
UF6	30	3.7027e+0 (8.18e-1) =	5.1808e+0 (5.98e-1) -	4.1460e+0 (7.30e-1) =	4.9152e+0 (5.39e-1) -	4.8268e+0 (5.51e-1) -	<b>3.4377e+0 (9.19e-1)</b>
	50	5.4036e+0 (5.57e-1) -	5.6604e+0 (4.16e-1) -	5.8037e+0 (5.75e-1) -	5.4199e+0 (2.44e-1) -	6.0253e+0 (4.20e-1) -	<b>4.1321e+0 (7.49e-1)</b>
	70	5.4699e+0 (4.08e-1) -	6.2370e+0 (4.31e-1) -	6.3011e+0 (3.49e-1) -	5.8918e+0 (2.54e-1) -	6.1732e+0 (2.52e-1) -	<b>4.6443e+0 (6.85e-1)</b>
	100	6.1905e+0 (3.71e-1) -	6.6300e+0 (3.28e-1) -	6.6689e+0 (3.55e-1) -	6.0061e+0 (2.42e-1) -	6.4164e+0 (3.32e-1) -	<b>4.6440e+0 (4.54e-1)</b>
UF7	30	<b>8.2892e-1 (1.63e-1) =</b>	1.2301e+0 (9.31e-2) -	1.0200e+0 (2.27e-1) =	1.2518e+0 (5.03e-2) -	1.0674e+0 (2.42e-1) -	8.6801e-1 (1.46e-1)
	50	1.3493e+0 (9.07e-2) -	1.4225e+0 (7.45e-2) -	1.4271e+0 (1.08e-1) -	1.3357e+0 (8.91e-2) -	1.3618e+0 (1.24e-1) -	<b>1.1737e+0 (9.11e-2)</b>
	70	1.4350e+0 (9.15e-2) -	1.5783e+0 (6.21e-2) -	1.5673e+0 (5.85e-2) -	1.4571e+0 (6.71e-2) -	1.5441e+0 (6.08e-2) -	<b>1.2265e+0 (1.51e-2)</b>
	100	1.5038e+0 (9.44e-2) -	1.6598e+0 (6.27e-2) -	1.7141e+0 (4.45e-2) -	1.5858e+0 (2.51e-2) -	1.6355e+0 (7.50e-2) -	<b>1.3223e+0 (8.9e-2)</b>
UF8	30	<b>3.5530e-1 (4.43e-2) =</b>	1.0607e+0 (2.72e-1) -	3.9478e-1 (2.25e-2) =	7.6149e-1 (6.18e-2) -	9.6572e-1 (4.43e-1) -	3.6567e-1 (3.91e-2)
	50	2.3594e+0 (6.85e-1) -	3.1104e+0 (1.51e-1) -	3.3064e+0 (5.02e-1) -	1.3964e+0 (1.67e-1) -	3.1590e+0 (2.71e-1) -	<b>4.4216e-1 (5.41e-2)</b>
	70	2.9164e+0 (3.49e-1) -	3.4534e+0 (1.82e-1) -	3.3481e+0 (1.53e-1) -	1.6153e+0 (2.10e-1) -	3.2812e+0 (1.53e-1) -	<b>4.5163e-1 (2.30e-2)</b>
	100	3.1322e+0 (3.06e-1) -	3.4897e+0 (1.69e-1) -	3.5893e+0 (1.25e-1) -	2.0228e+0 (3.46e-1) -	3.5116e+0 (1.59e-1) -	<b>5.5226e-1 (5.38e-2)</b>
UF9	30	8.4365e-1 (4.17e-2) =	1.1358e+0 (2.19e-1) -	<b>7.5258e-1 (6.31e-2) =</b>	1.0795e+0 (1.42e-1) -	1.3157e+0 (8.60e-1) -	8.4367e-1 (2.16e-2)
	50	2.5051e+0 (7.71e-1) -	3.1280e+0 (4.31e-1) -	2.9989e+0 (7.18e-1) -	1.2332e+0 (2.63e-1) -	3.2526e+0 (2.29e-1) -	<b>8.7750e-1 (1.30e-2)</b>
	70	2.9139e+0 (3.08e-1) -	3.4978e+0 (1.65e-1) -	3.3717e+0 (1.80e-1) -	1.6837e+0 (1.50e-1) -	3.3328e+0 (2.39e-1) -	<b>8.7486e-1 (1.57e-2)</b>
	100	3.1881e+0 (2.72e-1) -	3.6452e+0 (1.15e-1) -	3.6681e+0 (1.22e-1) -	2.0393e+0 (2.36e-1) -	3.7383e+0 (9.14e-2) -	<b>8.8572e-1 (1.48e-2)</b>
MaF1	30	1.4595e-1 (1.90e-2) -	6.8729e-1 (5.35e-2) -	2.3197e-1 (2.92e-2) -	4.0123e-1 (3.23e-2) -	8.6181e-1 (2.89e-1) -	<b>9.4963e-2 (1.94e-2)</b>
	50	2.3283e+0 (3.53e-1) -	2.8252e+0 (5.03e-1) -	2.7151e+0 (3.63e-1) -	9.7500e-1 (1.53e-1) -	3.0011e+0 (2.65e-1) -	<b>1.1737e+0 (6.19e-2)</b>
	70	3.4332e+0 (4.87e-1) -	3.6044e+0 (1.17e+0) -	4.7443e+0 (3.28e-1) -	2.0461e+0 (2.53e-1) -	4.4697e+0 (4.20e-1) -	<b>3.9776e-1 (1.21e-1)</b>
	100	5.9599e+0 (7.35e-1) -	6.9358e+0 (5.24e-1) -	7.1025e+0 (2.95e-1) -	3.0450e+0 (1.53e-1) -	6.8168e+0 (3.83e-1) -	<b>9.0010e-1 (2.46e-1)</b>
MaF2	30	<b>7.1502e-2 (6.77e-3) =</b>	1.1493e-1 (1.43e-2) -	9.6309e-2 (6.47e-3) -	8.8665e-2 (2.03e-3) =	1.4184e-1 (3.41e-3) -	7.6100e-2 (1.92e-2)
	50	2.1507e-1 (1.13e-2) -	2.3752e-1 (6.05e-3) -	2.3332e-1 (8.50e-3) -	1.3612e-1 (7.25e-3) -	2.3771e-1 (8.27e-3) -	<b>9.9196e-2 (1.28e-2)</b>
	70	3.1320e-1 (1.51e-2) -	3.4352e-1 (1.20e-2) -	3.4678e-1 (9.13e-3) -	1.9653e-1 (3.80e-3) -	3.4587e-1 (7.96e-3) -	<b>1.1752e-1 (1.91e-2)</b>
	100	4.8651e-1 (1.89e-2) -	5.0762e-1 (1.89e-2) -	5.0537e-1 (1.66e-2) -	2.8713e-1 (8.58e-4) -	5.0581e-1 (1.76e-2) -	<b>1.8808e-1 (2.81e-2)</b>
MaF3	30	3.5258e+6 (7.17e+5) =	4.3157e+6 (1.15e+6) -	5.9072e+6 (1.42e+6) -	4.7645e+6 (7.34e+5) -	3.4844e+6 (5.37e+5) =	<b>3.3657e+6 (3.67e+5)</b>
	50	1.7687e+7 (3.39e+6) -	2.1395e+7 (3.04e+6) -	2.2766e+7 (2.44e+6) -	1.7216e+7 (1.51e+6) -	2.2075e+7 (2.99e+6) -	<b>1.2841e+7 (1.37e+6)</b>
	70	4.0193e+7 (7.89e+6) -	3.9885e+7 (6.25e+6) -	4.4230e+7 (5.60e+6) -	3.6489e+7 (4.24e+6) -	4.1501e+7 (3.43e+6) -	<b>2.9315e+7 (1.92e+6)</b>
	100	7.9285e+7 (1.65e+7) -	9.3876e+7 (1.02e+7) -	9.4426e+7 (7.06e+6) -	7.5574e+7 (5.17e+6) -	9.5815e+7 (8.97e+6) -	<b>6.5479e+7 (4.96e+6)</b>
MaF4	30	5.1074e+3 (6.76e+2) =	<b>4.4160e+3 (9.22e+2) =</b>	6.1768e+3 (5.64e+2) =	6.3698e+3 (5.80e+2) =	6.3418e+3 (2.45e+2) -	5.5648e+3 (8.45e+2)
	50	<b>1.0333e+4 (1.10e+3) =</b>	1.1667e+4 (9.09e+2) -	1.1183e+4 (7.14e+2) -	1.1860e+4 (6.64e+2) -	1.1480e+4 (7.53e+2) -	<b>1.0708e+4 (6.29e+2)</b>
	70	1.6451e+4 (1.57e+3) =	1.6987e+4 (9.87e+2) =	1.7050e+4 (8.03e+2) =	1.7142e+4 (7.51e+2) =	1.7770e+4 (7.98e+2) =	<b>1.6295e+4 (9.27e+2)</b>
	100	<b>2.4059e+4 (1.60e+3) =</b>	2.5266e+4 (1.68e+3) =	2.5557e+4 (6.18e+2) =	2.6472e+4 (3.73e+2) =	2.5695e+4 (1.20e+3) =	<b>2.4665e+4 (1.66e+3)</b>
MaF5	30	3.7279e+0 (7.37e-1) =	5.9333e+0 (8.74e-1) -	4.3918e+0 (1.19e+0) -	6.2462e+0 (3.92e-1) -	3.8345e+0 (9.75e-1) =	<b>3.2530e+0 (7.86e-1)</b>
	50	8.1020e+0 (1.09e+0) =	9.5831e+0 (8.70e-1) -	9.0525e+0 (1.08e+0) -	7.9982e+0 (1.90e+0) -	9.3524e+0 (1.55e+0) -	<b>5.1353e+0 (1.11e+0)</b>
	70	1.1325e+1 (2.00e+0) =	1.2773e+1 (9.36e-1) -	1.2648e+1 (1.38e+0) -	1.1405e+1 (1.80e+0) -	1.2498e+1 (1.10e+0) -	<b>7.3261e+0 (8.73e-1)</b>
	100	1.4776e+1 (1.14e+0) =	1.7543e+1 (1.46e+0) -	1.7837e+1 (1.30e+0) -	1.7377e+1 (1.43e+0) -	1.6943e+1 (6.33e-1) -	<b>9.6278e+0 (1.18e+0)</b>
MaF6	30	1.2513e+1 (4.31e+0) =	4.2496e+1 (1.43e+1) -	2.4254e+1 (5.64e+0) -	2.4081e+1 (7.26e+0) -	6.8371e+1 (9.18e+0) -	<b>8.7198e+0 (4.57e+0)</b>
	50	1.8812e+2 (3.34e+1) -	2.5929e+2 (1.78e+1) -	2.4115e+2 (2.56e+1) -	6.4983e+1 (1.34e+1) -	2.3791e+2 (1.69e+1) -	<b>3.8636e+1 (1.32e+1)</b>
	70	3.1343e+2 (2.43e+1) -	3.6376e+2 (3.14e+1) -	3.6685e+2 (2.35e+1) -	1.6604e+2 (6.02e+0) -	3.6223e+2 (2.60e+1) -	<b>1.0348e+2 (2.66e+1)</b>
	100	4.9744e+2 (5.68e+1) -	5.6636e+2 (6.00e+1) -	5.7406e+2 (3.26e+1) -	<b>1.8845e+2 (4.46e+1) =</b>	5.5927e+2 (5.59e+1) -	<b>2.1881e+2 (2.93e+1)</b>
MaF7	30	9.6262e-1 (3.92e-1) =	1.2384e+0 (2.18e+0) =	1.0549e+0 (3.54e-1) =	5.8984e+0 (9.69e-1) =	7.8552e+0 (5.06e+0) =	<b>8.5116e-1 (3.68e-1)</b>
	50	8.4142e+0 (7.89e-1) -	9.2743e+0 (6.98e-1) -	8.3600e+0 (1.32e+0) -	7.0490e+0 (6.83e-1) -	9.1401e+0 (1.97e-1) -	<b>1.6273e+0 (7.28e-1)</b>
	70	8.9116e+0 (8.58e-1) -	9.5072e+0 (4.59e-1) -	9.6951e+0 (3.64e-1) -	8.1832e+0 (5.62e-1) -	9.6435e+0 (4.89e-1) -	<b>1.6543e+0 (1.03e+0)</b>
	100	9.5902e+0 (3.70e-1) -	9.7683e+0 (3.04e-1) -	9.8039e+0 (3.15e-1) -	9.0127e+0 (5.92e-1) -	1.0021e+1 (4.46e-1) -	<b>3.8288e+0 (7.07e-1)</b>
+/-=		0/46/18	1/59/4	1/52/11	1/51/3	0/60/4	

better on the remaining cases. HeE-MOEA can only achieve significantly better IGD results on 100-D MaF. In addition, K-RVEA, ESF-RVEA, and KTA2 are unable to show significantly better performance on any test case.

In order to examine the convergence speed of the compared algorithms, their mean IGD values versus the function evaluation numbers over 30 independent runs are plotted in Figure 5, where the UF and MaF test problems with 70 decision variables are used as the representative cases. It can be observed from Figure 5 that the convergence curve of the mean

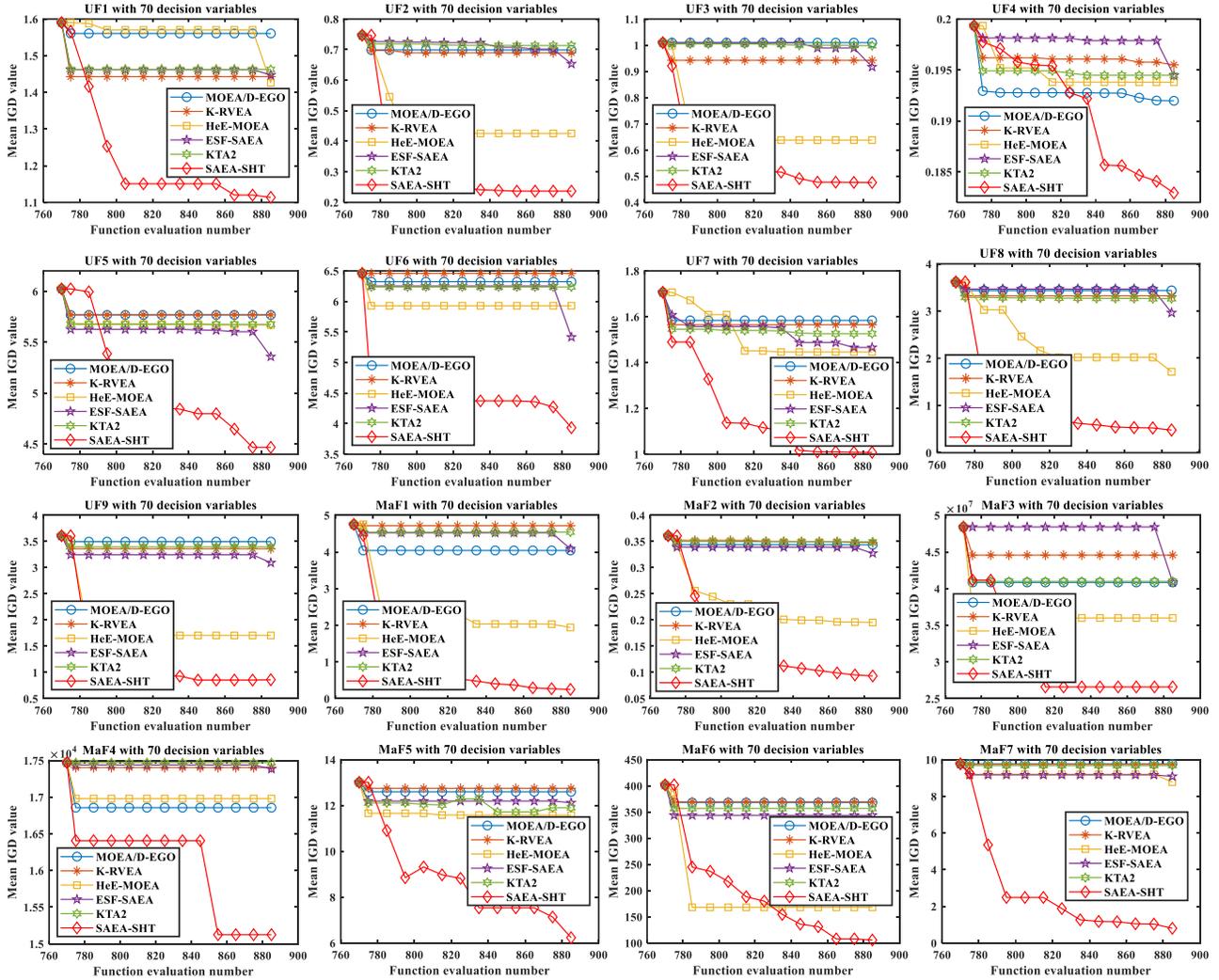


Figure 5: Convergence profiles of all the compared algorithms on the UF and MaF test problems with  $d = 70$

IGD values achieved by SAEA-SHT shows the fastest convergence speed on all 70-D UF and MaF test problems. The promising convergence speed of SAEA-SHT is mainly attributed to the generation of the helper task and the adopted SEMO.

To visually show the optimization performance, Figure A.2 of Appendix A plots the final nondominated solutions obtained by each compared algorithm with the median IGD values over 30 independent runs on several representative 70-D UF and MaF test problems. Figure A.2 of Appendix A shows that the final solutions of UF6 obtained by all the compared algorithms do not have good convergence to the true PF, while SAEA-SHT can achieve a better approximation than the other algorithms. For UF2 and UF4, which are plotted in Figure A.2 of Appendix A, the final solutions obtained by all the algorithms have good convergence but cannot be evenly spread over the whole PF. However, the final solutions obtained by SAEA-SHT still show better diversity than the other algorithms. When solving MaF2 in Figure A.2 of Appendix A,

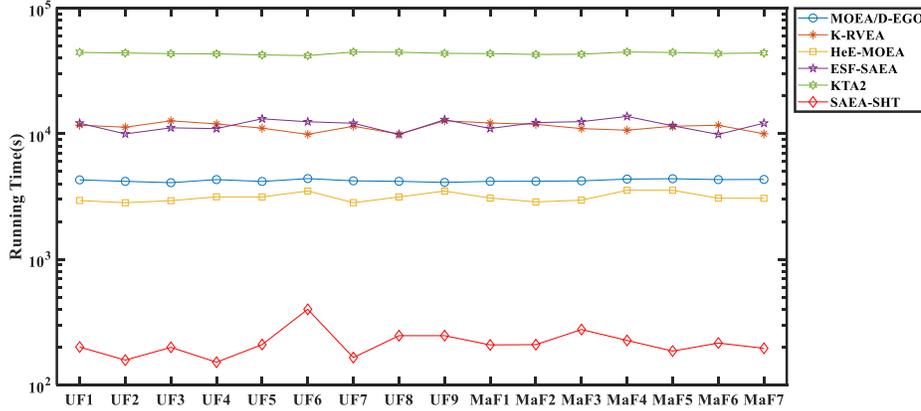


Figure 6: Average running times of all the compared algorithms on the UF and MaF test problems with  $d = 70$

SAEA-SHT also outperforms MOEA/D-EGO, K-RVEA, HeE-MOEA, ESF-RVEA, and KTA2 in terms of both convergence and diversity according to their final solutions.

#### 4.4 More Discussions

##### 4.4.1 Computational Time

To further evaluate the actual runtime of the six above SAEAs adopted in our comparative study, their average running times (in seconds: s) are plotted in Figure 6, where the UF and MaF test problems with 70 decision values are used as the representative cases. According to Figure 6, our proposed algorithm SAEA-SHT has a much lower computational complexity, mainly because SAEA-SHT adopts an incremental Kriging model as its surrogate model. HeE-MOEA and MOEA/D-EGO are ranked second and third, respectively, and their computational times are smaller than those of the other algorithms by at least one order of magnitude except for SAEA-SHT. In addition, the computational times of K-RVEA, ESF-RVEA, and KTA2 are considerably greater than those of SAEA-SHT, HeE-MOEA, and MOEA/D-EGO, which can be attributed to the use of Kriging models in these algorithms. The high computational cost restricts them from tackling high-dimensional optimization problems. Thus, our proposed algorithm SAEA-SHT not only achieves superior optimization performance, but also reduces the computational cost.

##### 4.4.2 Comparisons with More State-of-the-art SAEAs

More recently, there have been several state-of-the-art SAEAs designed for tackling high-dimensional EMOPs. In this section, two SAEAs (i.e., EDN-ARMOEA [Guo et al. 2022] and MCEA/D [Sonoda and Nakata 2022]) are included for performance comparison with SAEA-SHT for solving all the UF and MaF test problems. Due to page limitations, their IGD results are provided in Table A.2 of Appendix A. Table 4 summarizes their pairwise comparison results based on the IGD results in Table A.2 of Appendix A. Note that in Table 4, “Better”, “Worse”, and “Similar” indicate respectively the number of test problems in which the performance of SAEA-SHT is better than, worse than, and similar to those of EDN-ARMOEA and MCEA/D, while “Best” indicates the number of test problems in which the corresponding algorithm performs best.

As observed from Table 4, SAEA-SHT shows advantages for tackling the 30-D, 50-D, 70-D, 100-D UF and MaF test problems. To be specific, the performance of SAEA-SHT is better than that of EDN-ARMOEA and MCEA/D on 57 and

Table 4: Summary of Comparison of SAEA-SHT with EDN-ARMOEA and MCEA/D

SAEA-SHT vs.		EDN-ARMOEA	MCEA/D
UF and MaF ( $d = 30, 50, 70, 100$ )	Better	57	48
	Worse	0	8
	Similar	7	8
Best		0	12

Table 5: Summary of Comparison of SAEA-SHT with Different  $n_d$  Values

SAEA-SHT ( $n_d = 10$ ) vs.		$n_d = 5$	$n_d = 15$	$n_d = 20$
UF and MaF ( $d = 30, 50, 70, 100$ )	Better	14	18	11
	Worse	7	4	10
	Similar	43	42	43
Best		31	7	18

48 out of 64 cases, respectively, while EDN-ARMOEA and MCEA/D significantly outperform or are competitive with SAEA-SHT on only 7 and 16 cases, respectively. Moreover, EDN-ARMOEA even fails to achieve significantly better IGD results than SAEA-SHT on any of 64 cases. Therefore, the advantages of our proposed algorithm over other state-of-the-art SAEAs are validated for solving high-dimensional test EMOPs.

#### 4.4.3 Sensitivity Analysis of Parameter $n_d$

To analyze the sensitivity of the parameter  $n_d$  (the dimension of the helper task) in SAEA-SHT, some experiments are run in this section. Four SAEA-SHT variants with different  $n_d$  values from  $\{5, 10, 15, 20\}$  are experimentally compared with other parameters settings in Section 4.2. The IGD results on all the UF and MaF test problems with 30, 50, 70, and 100 decision variables are provided in Table A.3 of Appendix A. Table 5 summarizes the comparison of the IGD results of SAEA-SHT with different  $n_d$  values based on their IGD results.

As observed from Table 5, SAEA-SHT using  $n_d=10$  has statistically similar IGD results with respect to those using  $n_d = 5$ ,  $n_d = 15$ , and  $n_d = 20$  on 43, 42, and 43 out of 64 cases, respectively. It seems that SAEA-SHT is not as sensitive to the setting of parameter  $n_d$  when changing from 5 to 20. Nevertheless, SAEA-SHT using  $n_d = 10$  is the best among the four compared variants since it achieves the best IGD results on the largest number of test cases, i.e., 31 out of 64 cases. The other three compared variants using  $n_d = 5$ ,  $n_d = 15$ , and  $n_d = 20$  achieve the best IGD results on 7, 8, and 18 cases, respectively. Furthermore, SAEA-SHT using  $n_d = 10$  performs better than that using  $n_d = 5$ ,  $n_d = 15$ , and  $n_d = 20$  on 14, 18, and 11 cases. Therefore, in this article, the value of parameter  $n_d$  is set as 10.

In addition, to further study the effect of different settings of parameter  $n_d$  for helper task generation in Section 3.1, SAEA-SHT is also compared to its variants with four different settings of  $n_d$  (i.e.,  $n_d = 0.2 \times d$ ,  $n_d = 0.4 \times d$ ,  $n_d = 0.6 \times d$ , and  $n_d = 0.8 \times d$ ). Due to page limitations, the comparison results are provided in Table A.4 of Appendix A and Table 6 summarizes their pairwise comparison results. The results reported in Table 6 show that SAEA-SHT with  $n_d = 10$  obtains the best results on 36 out of 64 cases, while other SAEA-SHT variants with different  $n_d$  values achieve the best results on the remaining 28 cases. As indicated by Wilcoxon's rank-sum test, SAEA-SHT with  $n_d = 10$  can perform significantly better than or similarly to SAEA-SHT with  $n_d = 0.2 \times d$ ,  $n_d = 0.4 \times d$ ,  $n_d = 0.6 \times d$ , and  $n_d = 0.8 \times d$  on 59, 60, 60, and 63 cases, respectively. Furthermore, SAEA-SHT with  $n_d = 10$  is outperformed by that with  $n_d = 0.2 \times d$ ,  $n_d = 0.4 \times d$ ,  $n_d = 0.6 \times d$ , and  $n_d = 0.8 \times d$  only on 5, 4, 4, and 1 cases, respectively. In addition,  $n_d = 0.2 \times d$  is the best algorithm among four variants, since it is able to show similar performance to  $n_d = 10$  on 51 out of 64 cases. Nevertheless, SAEA-SHT with  $n_d = 0.2 \times d$  performs significantly better than that with  $n_d=10$  on only 5 out of 64 cases. Furthermore, Figure 7 plots the average computational time consumed to complete one trial for all the UF and MaF test problems with  $d = 100$ . From the table, it

Table 6: Summary of Comparison of SAEA-SHT with Different  $n_d$  Values

SAEA-SHT ( $n_d = 10$ ) vs.		$n_d = 0.2 \times d$	$n_d = 0.4 \times d$	$n_d = 0.6 \times d$	$n_d = 0.8 \times d$	
UF and MaF ( $d = 30, 50, 70, 100$ )	Better	8	16	28	32	
	Worse	5	4	4	1	
	Similar	51	44	32	31	
Best		36	14	8	4	2

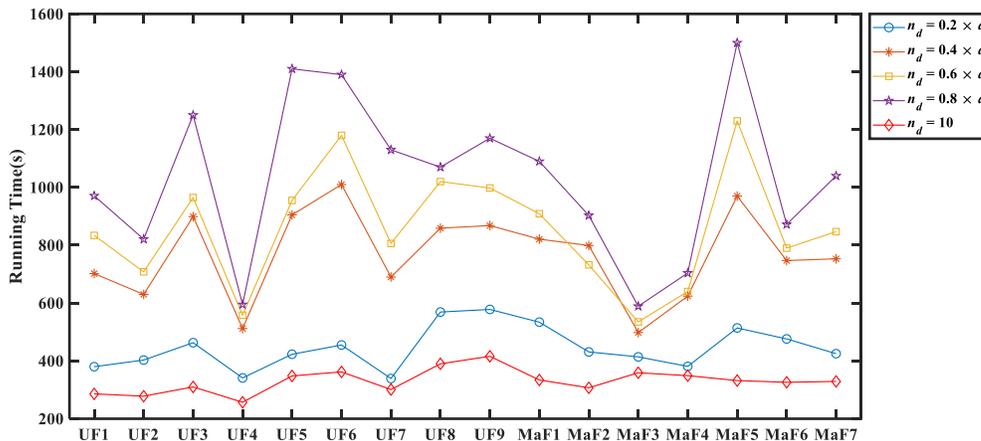


Figure 7: Average running times of five compared algorithms on the UF and MaF test problems with  $d = 100$

is evident that  $n_d = 10$  performs much faster than the other four variants, as it builds computationally efficient Kriging models. It can be also observed that the computational time of SAEA-SHT is highly dependent on the increase of  $n_d$ , since the computational complexity of the employed Kriging models mainly depends on the dimensionality of the training samples. As a consequence, in consideration of the optimization performance and computational efficiency,  $n_d = 10$  is recommended as the value of parameter  $n_d$ .

#### 4.4.4 Effectiveness of the Helper Task Generation

To study the effectiveness of the helper task generation introduced in Section 3.1, more experiments are executed to compare SAEA-SHT with its variant without using the helper task generation (called SAEA-SHT-I). Thus, SAEA-SHT-I only performs a general SAEA without the assistance of the helper task, which does not use the evolutionary multitasking optimizer. Their IGD results are presented in Table A.5 of Appendix A. Table 7 summarizes their pairwise comparison results. It can be observed from Table 7 that SAEA-SHT shows distinct advantages when tackling 30-D, 50-D, 70-D, and 100-D UF and MaF test problems. In addition, SAEA-SHT performs significantly better than or similarly to SAEA-SHT-I on 50 out of 64 cases, while SAEA-SHT-I only performs better on 14 cases. Therefore, the proposed helper task generation is validated to have a significant contribution to the performance of the SAEA-SHT.

Moreover, in order to observe the effectiveness of the knowledge transfer between two tasks more clearly, the average transfer successful rate from the helper task to the target task on DTLZ3, WFG9, and MaF3 with 70 decision variables on 30 runs is plotted in Figure 8. Note that for test problems with  $d = 70$ , there are  $11 \times d + 119 = 889$  function evaluations that can be used for evolutionary optimization, and  $11 \times d - 1 = 769$  function evaluations are consumed to generate training samples. Thus, the knowledge transfer from the helper task to the target task takes place from 770 to 889 function evaluations (in a total of 120 function evaluations). As shown from Figure 8, for DTLZ3, the average transfer successful rate is maintained at low and high levels in the former 80 function evaluations and the latter 40 function evaluations, respectively. Since

Table 7: Summary of Comparison of SAEA-SHT with Six Variants

SAEA-SHT vs.		SAEA-SHT-I	ExI	LCB	PoI	SAEA-SHT-E	SAEA-SHT-U
UF and MaF ( $d = 30, 50, 70, 100$ )	Better	42	43	45	49	38	44
	Worse	14	4	1	0	11	5
	Similar	8	17	18	15	15	15
Best		44	9	2	1	0	2

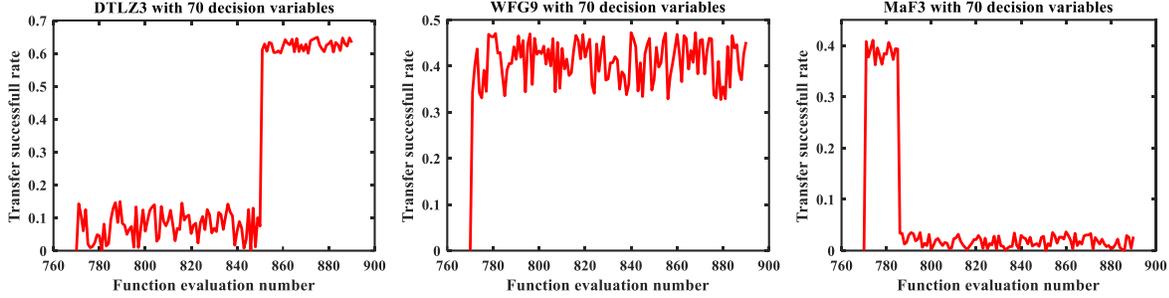


Figure 8: Average transfer successful rate from the helper task to the target task on DTLZ3, WFG9, MaF3 with  $d = 70$

population in the helper task may be far away from the true PF of the target problem, the transferred solutions generated from knowledge transfer are discarded by the target task. Nevertheless, after the transferred population converges to the true PF, the knowledge transfer is capable of assisting the optimization of the target task. Thus, since the search experiences of the helper task are helpful for the target task, the successful rate of knowledge transfer is maintained at high level. As for WFG9, the successful transfer rate is maintained at around 0.4 during the entire evolutionary process, which indicates that the helper task is able to provide beneficial knowledge for the target task from beginning to end. With respect to MaF3, the successful transfer rate is maintained at a very low level in the later evolutionary process, however, it is always larger than 0. Although the auxiliary efficiency is very low, the target task can take advantage of the successfully transferred solutions to generate more nondominated solutions. Therefore, the above experimental results fully demonstrate the effectiveness of the knowledge transfer from the helper task to the target task.

#### 4.4.5 Effectiveness of the Model Management Strategy

To further verify the effectiveness of the proposed model management strategy introduced in Section 3.4, three popular model management strategies (ExI, LCB, PoI) are embedded into SAEA-SHT, giving rise to three SAEA-SHT variants identified as ExI, LCB, and PoI in this article for the sake of simplicity. Their IGD results are presented in Table A.5 of Appendix A for tackling 30-D, 50-D, 70-D, and 100-D UF and MaF test problems. Table 7 summarizes the pairwise comparison results of SAEA-SHT with ExI, LCB, and PoI. As observed from Table 7, our model management strategy in SAEA-SHT shows superior optimization performance over other strategies, as SAEA-SHT is better than ExI, LCB, and PoI on 43, 45, and 49 out of 64 cases, respectively, while only ExI and LCB perform better than our strategy on 4 and 1 cases, respectively. Since the three model management strategies were originally designed for solving expensive single-objective optimization problems, it is reasonable that they cannot perform as well in SAEA-SHT because SAEA-SHT does not decompose high-dimensional EMOPs into single-objective optimization problems. Thus, the model management strategy proposed in SAEA-SHT is more effective, as validated by these experiments.

Moreover, since the proposed model management strategy selects new samples with the minimum Euclidean distance or maximum uncertainty, SAEA-SHT is further compared with the other two variants:

- SAEA-SHT-E only selects solutions with the minimum Euclidean distance.

- SAEA-SHT-U only selects solutions with the maximum uncertainty.

The experiments are conducted on UF and MaF test problems with 30, 50, 70, and 100 decision variables. Table A.5 of Appendix A provides the IGD results yielded by SAEA-SHT, SAEA-SHT-E, and SAEA-SHT-U over 30 independent runs. Table 7 gives the pairwise comparison results of SAEA-SHT, SAEA-SHT-E, and SAEA-SHT-U. As observed from Table 7, SAEA-SHT obtains the best IGD results on more than half of the cases, i.e., on 44 out of 64 cases, which confirms that SAEA-SHT is the best one in the comparison. The two variants SAEA-SHT-E and SAEA-SHT-U obtain the best IGD results on 6 and 2 test cases, respectively. Thus, the above experimental results and analyses validate the effectiveness of the proposed model management strategy.

## 5 CONCLUSIONS AND FUTURE WORK

In this article, we have proposed a novel surrogate-assisted evolutionary algorithm with a simplified helper task, called SAEA-SHT, for high-dimensional expensive multiobjective optimization. SAEA-SHT consists of three main components, i.e., surrogate training for the tasks, a surrogate-assisted evolutionary multitasking optimizer, and a model management strategy. In surrogate training for the tasks, a helper task including only some of the decision variables is constructed to help tackle the target task (the target EMOPs). After that, one incremental surrogate model is trained with gradually evaluated samples for the target task, and one general surrogate model is trained at each generation for the helper task to enhance the diversity of knowledge transfer. Then, a surrogate-assisted evolutionary multitasking optimizer is designed to optimize the target task with the assistance of a simplified helper task. Finally, an effective model management strategy is proposed to identify a few new samples to improve the prediction accuracy of surrogate models by ensuring both the convergence and diversity of the training data. When compared with five competitive SAEAs, the experiments on four test suites (DTLZ, WFG, UF, and MaF) with up to 100 dimensions have confirmed the advantages of our algorithm in most cases. Moreover, the sensitivity analysis of parameters, the effectiveness of the helper task generation, and the effectiveness of the model management strategy are experimentally studied.

In this paper, although the random dimensionality reduction technique is successfully used to generate a simplified helper task in an artificial way, its theoretical analysis is not included in this paper, but we plan to undertake this task as part of our future work. In addition, the performance of SAEA-SHT on EMOPs with more than 100 dimensions will be further studied. Moreover, the application of SAEA-SHT to some real-world applications will also be considered in our future work.

## REFERENCES

- George De Ath, Richard M Everson, Alma AM Rahat, and Jonathan E Fieldsend. 2021. Greed is good: Exploration and exploitation trade-offs in Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*. 1, 1 (2021), 1–22.
- Mickael Binois and Nathan Wycoff. 2022. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*. 2, 2 (2022), 1–26.
- Nicola Beume, Boris Naujoks, and Michael Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*. 181, 3 (2007), 1653–1669.
- Ke Chen, Bing Xue, Mengjie Zhang, and Fengyu Zhou. 2021. Evolutionary Multitasking for Feature Selection in High-Dimensional Classification via Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*. 26, 3 (2021), 446–460.
- Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2016. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*. 20, 5 (2016), 773–791.
- Ran Cheng, Miqing Li, Ye Tian, Xingyi Zhang, Shengxiang Yang, Yaochu Jin, and Xin Yao. 2017. A benchmark test suite for evolutionary many-objective optimization. *Complex & Intelligent Systems*. 3 (2017), 67–81.
- Tinkle Chugh, Nirupam Chakraborti, Karthik Sindhya, and Yaochu Jin. 2017. A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem. *Materials and Manufacturing Processes*. 32, 10 (2017), 1172–1178.
- Tinkle Chugh, Yaochu Jin, Kaisa Miettinen, Jussi Hakanen, and Karthik Sindhya. 2018. A Surrogate-Assisted Reference Vector Guided Evolutionary

- Algorithm for Computationally Expensive Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*. 22, 1 (2018), 129–142.
- Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computation*. 23, 9 (2019), 3137–3166.
- Carlos A. Coello Coello and Margarita Reyes Sierra. 2004. A study of the parallelization of a coevolutionary multiobjective evolutionary algorithm. in *Artificial Intelligence: Third Mexican International Conference on Artificial Intelligence*. (2004), 688–697.
- Kalyanmoy Deb and Ram Bhushan Agrawal. 1995. Simulated binary crossover for continuous search space. *Complex Systems*. 9, 2 (1995), 115–148.
- Kalyanmoy Deb and Mayank Goyal. 1996. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and informatics*. 26, 4 (1996), 30–45.
- Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 6, 2 (2002), 182–197.
- Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2002. Scalable multiobjective optimization test problems. in *Proceedings of the 2002 Congress on Evolutionary Computation*. 1 (2002), 825–830.
- Dawei Ding, Jing Xia, Lixia Yang, and Xiaodong Ding. 2018. Multiobjective Optimization Design for Electrically Large Coverage: Fragment-Type NearField/Far-Field UHF RFID Reader Antenna Design. *IEEE Antennas and Propagation Magazine*. 60, 1 (2018), 27–37.
- Michael Emmerich, Nicola Beume, and Boris Naujoks. 2005. An EMO algorithm using the hypervolume measure as selection criterion. in *International Conference on Evolutionary Multi-Criterion Optimization*. (2005), 62–76.
- Michael Emmerich, Kyriakos C. Giannakoglou, and Boris Naujoks. 2006. Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*. 10, 4 (2006), 421–439.
- Michael Emmerich, Alexios Giotis, Mutlu Özdemir, Thomas Back, and Kyriakos Giannakoglou. 2002. Metamodel-Assisted evolution strategies. in *International Conference on Parallel Problem Solving from Nature*. (2002), 361–370.
- Yinglan Feng, Liang Feng, Yaqing Hou, Kay Chen Tan, and Sam Kwong. 2021. EMT-ReMO: Evolutionary Multitasking for High-Dimensional Multi-Objective Optimization via Random Embedding. in *2021 IEEE Congress on Evolutionary Computation (CEC)*. (2021), 1672-1679.
- Dan Guo, Tianyou Chai, Jinliang Ding, and Yaochu Jin. 2016. Small data driven evolutionary multiobjective optimization of fused magnesium furnaces. in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. (2016), 1–8.
- Dan Guo, Yaochu Jin, Jinliang Ding, and Tianyou Chai. 2019. Heterogeneous ensemble-based infill criterion for evolutionary multiobjective optimization of expensive problems. *IEEE Transactions on Cybernetics*. 49, 3 (2019), 1012–1025.
- Dan Guo, Xilu Wang, Kailai Gao, Yaochu Jin, Jinliang Ding, and Tianyou Chai. 2022. Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 52, 4 (2022), 2084-2097.
- Daofu Guo, Zhigang Ren, Yongsheng Liang, and An Chen. 2000. Scaling up radial basis function for high-dimensional expensive optimization using random projection. in *2020 IEEE Congress on Evolutionary Computation (CEC)*. (2000), 1-8.
- Abhishek Gupta, Yew-Soon Ong, and Liang Feng. 2015. Multifactorial evolution: toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*. 20, 3 (2015), 343–357.
- Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. 2016. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Transactions on Cybernetics*. 47, 7 (2016), 1652–1665.
- Ahsanul Habib, Hemant Kumar Singh, Tinkle Chugh, Tapabrata Ray, and Kaisa Miettinen. 2019. A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive multi/many-objective optimization. *IEEE Transactions on Evolutionary Computation*. 23, 6 (2019), 1000–1014.
- Manuel Herrera, Aurore Guglielmi, Manyu Xiao, and Rajan Filomeno Coelho. 2014. Metamodelassisted optimization based on multiple kernel regression for mixed variables. *Structural and multidisciplinary optimization*. 49, 6 (2014), 979–991.
- Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. 2006. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*. 10, 5 (2006), 477–506.
- Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*. 1, 2 (2011), 61–70.
- Yaochu Jin and Bernhard Sendhoff. 2004. Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles. in *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*. (2004), 688–699.
- Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. 2018. Data-driven evolutionary optimization: an overview and case studies. *IEEE Transactions on Evolutionary Computation*. 23, 3 (2018), 442–458.
- Hao Jiang, Yuhang Wang, Ye Tian, Xingyi Zhang, and Jianhua Xiao. 2021. Feature construction for meta-heuristic algorithm recommendation of capacitated vehicle routing problems. *ACM Transactions on Evolutionary Learning and Optimization*. 1, 1 (2021), 1–28.
- Donald R Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*. 13, 4 (1998), 455–492.
- Nicolas Jozefowicz, Frédéric Semet, and El-Ghazali Talbi. 2021. Multiobjective vehicle routing problems. *European Journal of Operational Research*. 189, 2 (2008), 293–309.
- Joshua Knowles. 2006. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*. 10, 1 (2006), 50–66.
- Weijian Kong, Tianyou Chai, Shengxiang Yang, and Jinliang Ding. 2013. A hybrid evolutionary multiobjective optimization strategy for the dynamic power

- supply problem in magnesia grain manufacturing. *Applied Soft Computing*. 13, 5 (2013), 2960–2969.
- Ke Li, Qingfu Zhang, Sam Kwong, Miqing Li, and Ran Wang. 2014. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*. 18, 6 (2014), 909–923.
- Jianqing Lin, Cheng He, and Ran Cheng. 2022. Adaptive dropout for high-dimensional expensive multiobjective optimization. *Complex & Intelligent Systems*. 8, 1 (2022), 271–285.
- Qiuzhen Lin, Xunfeng Wu, Lijia Ma, Jianqiang Li, Maoguo Gong, and Carlos A. Coello Coello. 2021. An ensemble surrogate-based framework for expensive multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*. 26, 4 (2021), 631–645.
- Hai-Lin Liu, Fangqing Gu, and Qingfu Zhang. 2014. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation*. 18, 3 (2014), 450–455.
- Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. 2010. Dominance-based Pareto surrogate for multiobjective optimization. in *Simulated Evolution and Learning: 8th International Conference*. (2010), 230–239.
- Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: Neural architecture search using multiobjective genetic algorithm. in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. (2019), 419–427.
- James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1, 14 (1967), 281–297.
- Helena Mala-Jetmarova, Nargiz Sultanova, and Dragan Savic. 2017. Lost in optimization of water distribution systems? a literature review of system operation. *Environmental Modelling and Software*. 93 (2017), 209–254.
- Michael D. McKay, Richard J. Beckman, and William J. Conover. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*. 42, 1 (2000), 55–61.
- Yew-Soon Ong and Abhishek Gupta. 2016. Evolutionary multitasking: A computer science view of cognitive multitasking. *Cognitive Computation*. 8, 2 (2016), 125–142.
- Linqiang Pan, Cheng He, Ye Tian, Handing Wang, Xingyi Zhang, and Yaochu Jin. 2019. A classification-based surrogate-assisted evolutionary algorithm for expensive manyobjective optimization. *IEEE Transactions on Evolutionary Computation*. 23, 1 (2019), 74–88.
- Wolfgang Ponweiser, Tobias wagner, Dirk Biermann, and Markus Vincze. 2008. Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection. in *International Conference on Parallel Problem Solving from Nature*. (2008), 784–794.
- Kangjia Qiao, Kunjie Yu, Boyang Qu, Jing Liang, Hui Song, Caitong Yue, Hongyu Lin, and Kay Chen Tan. 2022. Dynamic Auxiliary Task-Based Evolutionary Multitasking for Constrained Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*. 27, 3 (2022), 642–656.
- Alma AM Rahat, Richard M. Everson, and Jonathan E. Fieldsend. 2017. Alternative infill strategies for expensive multiobjective optimization. in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. (2017), 873–880.
- R. Saravanan, S. Ramabalan, N. Godwin Raja Ebenezer, and C. Dharmaraja. 2009. Evolutionary multi criteria design optimization of robot grippers. *Applied Soft Computing*. 9, 1 (2009), 159–172.
- Takum Sonoda and Masaya Nakata. 2022. Multiple Classifiers-Assisted Evolutionary Algorithm Based on Decomposition for High-Dimensional Multiobjective Problems. *IEEE Transactions on Evolutionary Computation*. 26, 6 (2022), 1581–1595.
- Zhenshou Song, Handing Wang, Cheng He, and Yaochu Jin. 2021. A kriging-assisted two-archive evolutionary algorithm for expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*. 25, 6 (2021), 1013–1027.
- Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G Yen, and Mengjie Zhang. 2020. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*. 24, 2 (2020), 350–364.
- Chaoli Sun, Yaochu Jin, Ran Cheng, Jinliang Ding, and Jianchao Zeng. 2017. Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems. *IEEE Transactions on Evolutionary Computation*. 21, 4 (2017), 644–660.
- Mohammad Tabatabaei, Markus Hartikainen, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. 2019. An interactive surrogate-based method for computationally expensive multiobjective optimization. *Journal of the Operational Research Society*. 70, 6 (2019), 898–914.
- Ye Tian, Ran Cheng, Xingyi Zhang, Fan Cheng, and Yaochu Jin. 2018. An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*. 22, 4 (2018), 609–622.
- Virginia Torczon and Michael Trosset. 1998. Using approximations to accelerate engineering design optimization. in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. (1998), 4800.
- Cuie Yang, Jinliang Ding, Yaochu Jin, and Tianyou Chai. 2019. Offline data-driven multiobjective optimization: Knowledge transfer between surrogates and generation of final solutions. *IEEE Transactions on Evolutionary Computation*. 24, 3 (2019), 409–423.
- Dawei Zhan and Huanlai Xing. 2021. A fast Kriging-assisted evolutionary algorithm based on incremental learning. *IEEE Transactions on Evolutionary Computation*. 25, 5 (2021), 941–955.
- Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*. 11, 6 (2007), 712–731.
- Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. 2010. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*. 14, 3 (2010), 456–474.
- Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. 2008. Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-objective Optimization Algorithms, Technical Report. (2008), 1–30.

- Zongzhao Zhou, Yew Soon Ong, My Hanh Nguyen, and Dudy Lim. 2005. A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. in 2005 IEEE Congress on Evolutionary Computation. 3 (2005), 2832–2839.
- Eckart Zitzler and Simon Künzli. 2004. Indicator-based selection in multiobjective search. in PPSN. 4 (2004), 832–842.
- Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report. 103 (2001).

APPENDIX

A SUPPLEMENTARY TABLES AND FIGURES

Table A.1: Features of the DTLZ, WFG, UF, and MaF test problems

Test Problems	m (Objectives)	D (Dimension)	Features
DTLZ1	3	30, 50, 70, 100	Linear, Multi-modal
DTLZ2	3	30, 50, 70, 100	Concave, Uni-modal
DTLZ3	3	30, 50, 70, 100	Concave, Multi-modal
DTLZ4	3	30, 50, 70, 100	Concave, Uni-modal
DTLZ5	3	30, 50, 70, 100	Degenerate
DTLZ6	3	30, 50, 70, 100	Degenerate
DTLZ7	3	30, 50, 70, 100	Mixed, Disconnected
WFG1	3	30, 50, 70, 100	Convex, Mixed
WFG2	3	30, 50, 70, 100	Convex, Disconnected, Mixed
WFG3	3	30, 50, 70, 100	Linear, Uni-modal
WFG4	3	30, 50, 70, 100	Concave, Deceptive
WFG5	3	30, 50, 70, 100	Concave, Multi-modal
WFG6	3	30, 50, 70, 100	Concave, Uni-modal
WFG7	3	30, 50, 70, 100	Concave, Uni-modal
WFG8	3	30, 50, 70, 100	Concave, Uni-modal
WFG9	3	30, 50, 70, 100	Concave, Multi-modal, Deceptive
UF1	2	30, 50, 70, 100	Convex, Multi-modal
UF2	2	30, 50, 70, 100	Convex, Multi-modal
UF3	2	30, 50, 70, 100	Convex, Multi-modal
UF4	2	30, 50, 70, 100	Concave, Multi-modal
UF5	2	30, 50, 70, 100	Linear, Disconnected
UF6	2	30, 50, 70, 100	Linear, Disconnected
UF7	2	30, 50, 70, 100	Linear, Multi-modal
UF8	3	30, 50, 70, 100	Concave, Multi-modal
UF9	3	30, 50, 70, 100	Mixed, Disconnected
MaF1	3	30, 50, 70, 100	Linear, Inverted
MaF2	3	30, 50, 70, 100	Concave
MaF3	3	30, 50, 70, 100	Convex, Multi-modal
MaF4	3	30, 50, 70, 100	Concave, Multi-modal, Inverted, Badly-scaled
MaF5	3	30, 50, 70, 100	Convex, Biased, Badly-scaled
MaF6	3	30, 50, 70, 100	Concave, Degenerate
MaF7	3	30, 50, 70, 100	Mixed, Disconnected, Multi-modal

Table A.2: Comparisons of SAEA-SHT with MCEA/D on the UF and MaF Test Suites

Problem	D	EDN-ARMOEA	MCEA/D	SAEA-SHT
UF1	30	1.1609e+0 (1.18e-1) -	1.0762e+0 (1.24e-1) -	<b>7.5239e-1 (1.04e-1)</b>
	50	1.3201e+0 (1.36e-1) =	1.3194e+0 (4.63e-2) -	<b>1.0632e+0 (1.17e-1)</b>
	70	1.5152e+0 (5.79e-2) -	1.3961e+0 (7.62e-2) -	<b>1.1534e+0 (1.30e-1)</b>
	100	1.6208e+0 (4.64e-2) -	1.4661e+0 (3.97e-2) -	<b>1.3029e+0 (1.33e-1)</b>
UF2	30	4.4068e-1 (6.08e-2) -	2.6963e-1 (1.30e-2) -	<b>1.7488e-1 (1.61e-2)</b>
	50	6.2392e-1 (1.67e-2) -	3.0128e-1 (1.92e-2) -	<b>2.1792e-1 (1.69e-2)</b>
	70	6.7565e-1 (2.94e-2) -	2.8590e-1 (1.36e-2) =	<b>2.5505e-1 (3.08e-2)</b>
	100	7.3251e-1 (1.04e-2) -	3.4223e-1 (1.89e-2) -	<b>2.8910e-1 (2.05e-2)</b>
UF3	30	9.3479e-1 (3.51e-2) -	7.1282e-1 (3.29e-2) -	<b>5.4848e-1 (2.42e-2)</b>
	50	9.0738e-1 (8.23e-2) -	6.2204e-1 (4.61e-2) -	<b>5.2924e-1 (3.81e-2)</b>
	70	8.8870e-1 (8.46e-2) -	5.5385e-1 (4.11e-2) =	<b>5.5071e-1 (4.27e-2)</b>
	100	9.6256e-1 (3.17e-2) -	<b>5.3112e-1 (2.83e-2)</b> +	6.0890e-1 (4.80e-2)
UF4	30	1.7785e-1 (5.14e-3) -	1.6901e-1 (5.79e-3) -	<b>1.4198e-1 (8.38e-3)</b>
	50	1.9061e-1 (5.51e-4) -	1.7994e-1 (5.05e-3) -	<b>1.7111e-1 (6.38e-3)</b>
	70	1.9505e-1 (2.81e-3) -	1.8892e-1 (4.58e-3) -	<b>1.8202e-1 (4.82e-3)</b>
	100	1.9881e-1 (1.68e-3) -	<b>1.9014e-1 (5.72e-3)</b> =	1.9259e-1 (2.44e-3)
UF5	30	4.7600e+0 (4.60e-1) =	4.9738e+0 (1.60e-1) -	<b>4.0302e+0 (4.15e-1)</b>
	50	5.4284e+0 (1.35e-1) -	5.3788e+0 (3.28e-1) -	<b>4.6321e+0 (4.68e-1)</b>
	70	5.7742e+0 (2.51e-1) -	5.7130e+0 (1.21e-1) -	<b>4.9570e+0 (3.30e-1)</b>
	100	5.8048e+0 (2.16e-1) -	5.5730e+0 (1.33e-1) -	<b>5.3280e+0 (3.03e-1)</b>
UF6	30	5.1010e+0 (5.09e-1) -	5.1045e+0 (3.80e-1) -	<b>3.4377e+0 (9.19e-1)</b>
	50	5.8289e+0 (2.77e-1) -	5.6412e+0 (3.79e-1) -	<b>4.1321e+0 (7.49e-1)</b>
	70	6.0304e+0 (4.15e-1) -	5.9510e+0 (3.78e-1) -	<b>4.6443e+0 (6.85e-1)</b>
	100	6.7577e+0 (3.49e-1) -	6.1764e+0 (2.72e-1) -	<b>5.3640e+0 (4.54e-1)</b>
UF7	30	1.1683e+0 (8.00e-2) -	1.1342e+0 (1.29e-1) -	<b>8.6801e-1 (1.46e-1)</b>
	50	1.4102e+0 (5.96e-2) -	1.3933e+0 (9.36e-2) -	<b>1.1737e+0 (9.11e-2)</b>
	70	1.4606e+0 (1.83e-1) =	1.4335e+0 (8.78e-2) =	<b>1.2565e+0 (1.51e-1)</b>
	100	1.6266e+0 (8.36e-2) -	1.5557e+0 (4.56e-2) -	<b>1.3223e+0 (8.96e-2)</b>
UF8	30	2.6218e+0 (1.90e-1) -	1.1595e+0 (2.15e-1) -	<b>3.6567e-1 (3.91e-2)</b>
	50	3.2529e+0 (1.23e-1) -	1.3082e+0 (1.71e-1) -	<b>4.4216e-1 (5.41e-2)</b>
	70	3.2892e+0 (2.01e-1) -	1.2742e+0 (1.47e-1) -	<b>4.5163e-1 (2.30e-2)</b>
	100	3.6366e+0 (1.22e-2) -	1.2670e+0 (1.71e-1) -	<b>5.5226e-1 (5.38e-2)</b>
UF9	30	2.5468e+0 (3.14e-1) -	1.1671e+0 (1.41e-1) -	<b>8.4367e-1 (2.16e-2)</b>
	50	3.0290e+0 (2.33e-1) -	1.1654e+0 (1.90e-1) -	<b>8.7750e-1 (1.30e-2)</b>
	70	3.2277e+0 (7.37e-2) -	1.1201e+0 (1.32e-1) -	<b>8.7486e-1 (1.57e-2)</b>
	100	3.6373e+0 (2.12e-1) -	1.3733e+0 (1.77e-1) -	<b>8.8572e-1 (1.48e-2)</b>
MaF1	30	1.5719e+0 (7.50e-2) -	4.7100e-1 (4.02e-2) -	<b>9.4963e-2 (1.94e-2)</b>
	50	2.9414e+0 (2.75e-1) -	6.2049e-1 (7.80e-2) -	<b>2.0126e-1 (6.19e-2)</b>
	70	4.6870e+0 (2.61e-1) -	8.7291e-1 (2.21e-1) -	<b>3.9776e-1 (1.21e-1)</b>
	100	7.1228e+0 (1.47e-1) -	1.4268e+0 (3.78e-1) -	<b>9.0010e-1 (2.46e-1)</b>
MaF2	30	1.4267e-1 (2.80e-3) -	9.4208e-2 (5.45e-3) -	<b>7.6100e-2 (1.92e-2)</b>
	50	2.4279e-1 (1.86e-3) -	1.4946e-1 (1.07e-2) -	<b>9.9196e-2 (1.28e-2)</b>
	70	3.5006e-1 (7.09e-3) -	1.7074e-1 (9.54e-3) -	<b>1.1752e-1 (1.91e-2)</b>
	100	5.0837e-1 (2.68e-3) -	2.2652e-1 (2.84e-2) -	<b>1.8808e-1 (2.81e-2)</b>
MaF3	30	6.3246e+6 (1.35e+6) -	<b>1.4987e+6 (9.35e+5)</b> +	3.3657e+6 (3.67e+5)
	50	2.0906e+7 (1.89e+6) -	<b>4.5716e+6 (2.69e+6)</b> +	1.2841e+7 (1.37e+6)
	70	4.3402e+7 (1.65e+6) -	<b>1.5190e+7 (7.75e+6)</b> +	2.9315e+7 (1.92e+6)
	100	9.8471e+7 (5.97e+6) -	<b>3.0566e+7 (1.63e+7)</b> +	6.5479e+7 (4.96e+6)
MaF4	30	6.1969e+3 (6.06e+2) =	<b>4.7405e+3 (1.22e+3)</b> =	5.5648e+3 (8.45e+2)
	50	1.1216e+4 (2.41e+2) =	<b>9.0050e+3 (2.33e+3)</b> =	1.0708e+4 (6.29e+2)
	70	1.7000e+4 (3.63e+2) =	<b>1.4045e+4 (1.03e+3)</b> +	1.6295e+4 (9.27e+2)
	100	2.5513e+4 (9.10e+2) =	<b>1.9333e+4 (4.08e+3)</b> +	2.4665e+4 (1.66e+3)
MaF5	30	5.6079e+0 (7.58e-1) -	5.1763e+0 (4.01e-1) -	<b>3.2530e+0 (7.86e-1)</b>
	50	9.3950e+0 (9.60e-1) -	7.1047e+0 (5.73e-1) -	<b>5.1353e+0 (1.11e+0)</b>
	70	1.2259e+1 (8.61e-1) -	8.6541e+0 (8.63e-1) -	<b>7.3261e+0 (8.73e-1)</b>
	100	1.7307e+1 (1.75e-1) -	1.2158e+1 (1.54e+0) -	<b>9.6278e+0 (1.18e+0)</b>
MaF6	30	1.2072e+2 (8.16e+0) -	2.6491e+1 (5.32e+0) -	<b>8.7198e+0 (4.57e+0)</b>
	50	2.3511e+2 (1.51e+1) -	5.1405e+1 (1.74e+1) =	<b>3.8636e+1 (1.32e+1)</b>
	70	3.8724e+2 (1.39e+1) -	<b>6.3287e+1 (8.37e+0)</b> =	1.0348e+2 (2.66e+1)
	100	5.8685e+2 (2.22e+1) -	<b>1.0565e+2 (3.72e+1)</b> +	2.1881e+2 (2.93e+1)
MaF7	30	3.6111e+0 (5.98e-1) -	8.1958e+0 (6.22e-1) -	<b>8.5116e-1 (3.68e-1)</b>
	50	4.8522e+0 (1.36e+0) -	9.3018e+0 (4.00e-1) -	<b>1.6273e+0 (7.28e-1)</b>
	70	5.7721e+0 (4.37e-1) -	9.3716e+0 (5.06e-1) -	<b>1.6543e+0 (1.03e+0)</b>
	100	6.6904e+0 (5.49e-1) -	9.8899e+0 (1.87e-1) -	<b>3.8288e+0 (7.07e-1)</b>
+/-/=		0/5/7/7	8/48/8	

Table A.3: Comparisons of SAEA-SHT with Different  $n_d$  Values on the UF and MaF Test Suites

Problem	D	$n_d=5$	$n_d=15$	$n_d=20$	SAEA-SHT ( $n_d=10$ )
UF1	30	<b>7.0046e-1 (1.56e-1)</b> +	7.4930e-1 (1.00e-1) =	7.0239e-1 (1.24e-1) +	7.5239e-1 (1.04e-1)
	50	<b>9.7608e-1 (1.32e-1)</b> +	9.9771e-1 (1.11e-1) =	9.8910e-1 (9.61e-2) +	1.0632e+0 (1.17e-1)
	70	1.0502e+0 (1.87e-1) =	1.0434e+0 (9.59e-2) =	<b>9.9154e-1 (9.77e-2)</b> +	1.1534e+0 (1.30e-1)
	100	<b>9.8692e-1 (1.26e-1)</b> +	1.0923e+0 (1.22e-1) +	1.0022e+0 (1.03e-1) +	1.3029e+0 (1.33e-1)
UF2	30	1.7408e-1 (1.60e-2) =	<b>1.6025e-1 (1.61e-2)</b> +	1.7108e-1 (1.11e-2) =	1.7488e-1 (1.61e-2)
	50	2.294e-1 (2.04e-2) =	<b>1.8992e-1 (2.22e-2)</b> +	2.0052e-1 (1.69e-2) =	2.1792e-1 (1.69e-2)
	70	2.6894e-1 (3.11e-2) =	<b>2.4578e-1 (3.18e-2)</b> +	2.5905e-1 (2.98e-2) =	2.5505e-1 (3.08e-2)
	100	2.8033e-1 (2.77e-2) =	<b>2.5676e-1 (2.17e-2)</b> +	2.9920e-1 (2.15e-2) =	2.8910e-1 (2.05e-2)
UF3	30	<b>5.2886e-1 (2.13e-2)</b> =	5.5048e-1 (2.12e-2) =	5.5840e-1 (1.22e-2) =	5.4848e-1 (2.42e-2)
	50	5.3324e-1 (3.29e-2) =	<b>5.1194e-1 (3.01e-2)</b> =	5.3304e-1 (2.91e-2) =	5.2924e-1 (3.81e-2)
	70	5.5631e-1 (4.28e-2) =	5.5291e-1 (3.37e-2) =	5.5471e-1 (4.01e-2) =	<b>5.5071e-1 (4.27e-2)</b>
	100	6.1667e-1 (4.00e-2) =	6.1890e-1 (4.08e-2) =	6.0903e-1 (2.00e-2) =	<b>6.0890e-1 (4.80e-2)</b>
UF4	30	1.4288e-1 (8.73e-3) =	1.5177e-1 (8.08e-3) =	<b>1.4008e-1 (9.08e-3)</b> =	1.4198e-1 (8.38e-3)
	50	1.8811e-1 (5.58e-3) =	1.9911e-1 (6.22e-3) =	<b>1.7011e-1 (5.28e-3)</b> =	1.7111e-1 (6.38e-3)
	70	1.9782e-1 (4.83e-3) =	1.9202e-1 (5.42e-3) =	1.8331e-1 (4.81e-3) =	<b>1.8202e-1 (4.82e-3)</b>
	100	2.3802e-1 (2.36e-3) =	2.2997e-1 (2.64e-3) =	1.9400e-1 (2.22e-3) =	<b>1.9259e-1 (2.44e-3)</b>
UF5	30	4.3318e+0 (4.53e-1) =	4.3345e+0 (4.48e-1) =	4.1892e+0 (4.25e-1) =	<b>4.0302e+0 (4.15e-1)</b>
	50	4.7873e+0 (4.55e-1) =	4.7754e+0 (4.44e-1) =	4.9921e+0 (4.00e-1) =	<b>4.6321e+0 (4.68e-1)</b>
	70	5.6898e+0 (3.20e-1) =	5.4570e+0 (3.30e-1) =	5.4290e+0 (3.20e-1) =	<b>4.9570e+0 (3.30e-1)</b>
	100	5.5306e+0 (2.87e-1) =	5.5580e+0 (2.03e-1) =	5.3680e+0 (3.88e-1) =	<b>5.3280e+0 (3.03e-1)</b>
UF6	30	3.1307e+0 (9.22e-1) +	3.4382e+0 (9.32e-1) =	<b>3.0031e+0 (7.01e-1)</b> +	3.4377e+0 (9.19e-1)
	50	4.0883e+0 (6.99e-1) =	4.2019e+0 (7.33e-1) =	<b>4.0129e+0 (6.98e-1)</b> +	4.1321e+0 (7.49e-1)
	70	<b>4.1127e+0 (4.95e-1)</b> +	4.4894e+0 (5.63e-1) =	4.3503e+0 (6.02e-1) +	4.6443e+0 (6.85e-1)
	100	<b>5.0129e+0 (3.34e-1)</b> +	5.3311e+0 (5.25e-1) =	5.0412e+0 (4.50e-1) +	5.3640e+0 (4.54e-1)
UF7	30	8.7001e-1 (1.26e-1) =	8.9833e-1 (1.22e-1) =	8.6901e-1 (1.06e-1) =	<b>8.6801e-1 (1.46e-1)</b>
	50	1.4712e+0 (9.81e-2) =	1.4030e+0 (9.29e-2) =	<b>1.0701e+0 (9.01e-2)</b> =	1.1737e+0 (9.11e-2)
	70	1.5522e+0 (1.52e-1) =	1.5021e+0 (1.44e-1) =	1.2905e+0 (1.00e-1) =	<b>1.2565e+0 (1.51e-1)</b>
	100	1.3799e+0 (8.72e-2) =	1.6423e+0 (7.26e-2) =	1.3338e+0 (8.99e-2) =	<b>1.3223e+0 (8.96e-2)</b>
UF8	30	3.6902e-1 (3.40e-2) =	3.6697e-1 (3.33e-2) =	<b>3.5500e-1 (3.91e-2)</b> =	3.6567e-1 (3.91e-2)
	50	4.6156e-1 (5.11e-2) =	4.5526e-1 (5.02e-2) =	4.5006e-1 (5.00e-2) =	<b>4.4216e-1 (5.41e-2)</b>
	70	4.7863e-1 (2.30e-2) =	4.7023e-1 (3.99e-2) =	4.5363e-1 (2.00e-2) =	<b>4.5163e-1 (2.30e-2)</b>
	100	5.6362e-1 (5.18e-2) =	5.6628e-1 (4.89e-2) =	<b>5.5208e-1 (4.44e-2)</b> =	5.5226e-1 (5.38e-2)
UF9	30	8.5367e-1 (2.29e-2) =	8.5067e-1 (2.66e-2) =	8.4722e-1 (1.82e-2) =	<b>8.4367e-1 (2.16e-2)</b>
	50	8.7701e-1 (1.32e-2) =	<b>8.7050e-1 (1.38e-2)</b> =	8.7956e-1 (1.78e-2) =	8.7750e-1 (1.30e-2)
	70	8.6982e-1 (1.55e-2) =	<b>8.6657e-1 (1.22e-2)</b> =	8.7406e-1 (1.51e-2) =	8.7486e-1 (1.57e-2)
	100	8.9702e-1 (1.33e-2) =	8.9903e-1 (1.83e-2) =	8.8662e-1 (1.98e-2) =	<b>8.8572e-1 (1.48e-2)</b>
MaF1	30	9.7136e-2 (1.34e-2) =	9.6102e-2 (2.20e-2) =	9.5191e-2 (1.88e-2) =	<b>9.4963e-2 (1.94e-2)</b>
	50	1.7948e-1 (5.69e-2) +	2.0226e-1 (6.10e-2) =	<b>1.7706e-1 (5.55e-2)</b> +	2.0126e-1 (6.19e-2)
	70	3.9776e-1 (1.81e-1) =	3.9006e-1 (1.13e-1) =	<b>3.8876e-1 (1.21e-1)</b> =	3.9776e-1 (1.21e-1)
	100	9.2258e-1 (2.62e-1) =	9.2210e-1 (2.16e-1) =	9.1189e-1 (2.02e-1) =	<b>9.0010e-1 (2.46e-1)</b>
MaF2	30	7.8956e-2 (1.88e-2) =	7.7755e-2 (1.23e-2) =	7.9120e-2 (1.90e-2) =	<b>7.6100e-2 (1.92e-2)</b>
	50	<b>8.3401e-2 (1.35e-2)</b> =	9.9656e-2 (1.33e-2) =	8.6196e-2 (2.21e-2) =	9.9196e-2 (1.28e-2)
	70	1.3354e-1 (1.92e-2) =	1.3002e-1 (1.05e-2) =	<b>1.0752e-1 (2.19e-2)</b> =	1.1752e-1 (1.91e-2)
	100	1.9848e-1 (3.34e-2) =	2.3088e-1 (2.36e-2) =	2.3408e-1 (2.77e-2) =	<b>1.8808e-1 (2.81e-2)</b>
MaF3	30	3.4575e+6 (3.12e+5) =	3.6632e+6 (4.47e+5) =	<b>3.0471e+6 (2.97e+5)</b> =	3.3657e+6 (3.67e+5)
	50	1.2928e+7 (1.07e+6) =	1.3819e+7 (1.97e+6) =	<b>1.0811e+7 (1.07e+6)</b> =	1.2841e+7 (1.37e+6)
	70	3.0515e+7 (2.96e+6) =	3.2105e+7 (1.77e+6) =	3.2108e+7 (1.99e+6) =	<b>2.9315e+7 (1.92e+6)</b>
	100	7.1797e+7 (4.88e+6) =	7.7794e+7 (4.62e+6) =	7.6011e+7 (4.55e+6) =	<b>6.5479e+7 (4.96e+6)</b>
MaF4	30	5.5648e+3 (8.22e+2) =	5.7707e+3 (8.35e+2) =	5.9624e+3 (7.87e+2) =	<b>5.5648e+3 (8.45e+2)</b>
	50	1.3708e+4 (6.10e+2) =	1.3378e+4 (7.59e+2) =	1.2722e+4 (6.33e+2) =	<b>1.0708e+4 (6.29e+2)</b>
	70	1.9295e+4 (8.71e+2) =	1.9975e+4 (7.47e+2) =	1.9711e+4 (8.28e+2) =	<b>1.6295e+4 (9.27e+2)</b>
	100	2.4665e+4 (1.79e+3) =	2.7054e+4 (2.05e+3) =	2.6625e+4 (1.73e+3) =	<b>2.4665e+4 (1.66e+3)</b>
MaF5	30	3.2790e+0 (7.34e-1) =	3.0534e+0 (7.09e-1) =	<b>3.0032e+0 (5.26e-1)</b> =	3.2530e+0 (7.86e-1)
	50	5.6307e+0 (1.23e+0) =	5.7722e+0 (1.79e+0) =	5.7353e+0 (2.18e+0) =	<b>5.1353e+0 (1.11e+0)</b>
	70	7.3579e+0 (8.85e-1) =	<b>7.2789e+0 (8.38e-1)</b> =	7.4422e+0 (9.23e-1) =	7.3261e+0 (8.73e-1)
	100	9.9838e+0 (1.30e+0) =	9.826e+0 (1.27e+0) =	9.9713e+0 (2.82e+0) =	<b>9.6278e+0 (1.18e+0)</b>
MaF6	30	8.4448e+0 (5.29e+0) =	8.5689e+0 (5.43e+0) =	<b>8.3332e+0 (5.50e+0)</b> =	8.7198e+0 (4.57e+0)
	50	3.5366e+1 (1.56e+1) =	3.6369e+1 (2.17e+1) =	<b>3.4033e+1 (1.77e+1)</b> =	3.8636e+1 (1.32e+1)
	70	1.1311e+2 (3.39e+1) =	1.1778e+2 (2.59e+1) =	<b>9.5508e+1 (2.12e+1)</b> +	1.0348e+2 (2.66e+1)
	100	2.3881e+2 (2.62e+1) =	2.2329e+2 (2.98e+1) =	<b>1.7872e+2 (2.60e+1)</b> +	2.1881e+2 (2.93e+1)
MaF7	30	1.3386e+0 (4.64e-1) =	1.5858e+0 (2.98e-1) =	1.5213e+0 (4.63e-1) =	<b>8.5116e-1 (3.68e-1)</b>
	50	2.4303e+0 (7.37e-1) =	2.5650e+0 (7.08e-1) =	2.3303e+0 (5.11e-1) =	<b>1.6273e+0 (7.28e-1)</b>
	70	3.8832e+0 (1.22e+0) =	4.1284e+0 (2.89e+0) =	3.1028e+0 (1.47e+0) =	<b>1.6543e+0 (1.03e+0)</b>
	100	6.1218e+0 (6.66e-1) =	5.9908e+0 (7.32e-1) =	5.0292e+0 (7.11e-1) =	<b>3.8288e+0 (7.07e-1)</b>
+/-/=		7/14/43	4/18/42	10/11/43	

Table A.4: Comparisons of SAEA-SHT with Different  $n_d$  Values on the UF and MaF Test Suites

Problem	D	$n_d = 0.2 \times d$	$n_d = 0.4 \times d$	$n_d = 0.6 \times d$	$n_d = 0.8 \times d$	SAEA-SHT ( $n_d = 10$ )
UF1	30	<b>7.0164e-1 (1.57e-1)</b> +	7.4273e-1 (1.14e-1) =	7.6841e-1 (9.39e-2) =	7.7753e-1 (1.64e-1) =	7.5239e-1 (1.04e-1)
	50	1.0444e+0 (1.60e-1) =	9.8097e-1 (8.34e-2) +	<b>9.7772e-1 (1.26e-1)</b> +	9.9988e-1 (1.46e-1) =	1.0632e+0 (1.17e-1)
	70	1.1247e+0 (1.45e-1) =	9.8618e-1 (1.41e-1) +	<b>9.7468e-1 (8.31e-2)</b> +	1.2280e+0 (9.32e-2) =	1.1534e+0 (1.30e-1)
	100	<b>1.0257e+0 (1.10e-1)</b> +	1.3021e+0 (1.45e-1) =	1.3266e+0 (1.47e-1) =	1.3901e+0 (1.20e-1) =	1.3029e+0 (1.33e-1)
UF2	30	1.7979e-1 (1.81e-2) =	<b>1.6130e-1 (1.49e-2)</b> +	1.6362e-1 (2.35e-2) +	1.8831e-1 (4.34e-2) -	1.7488e-1 (1.61e-2)
	50	<b>2.1725e-1 (4.57e-2)</b> =	2.2152e-1 (3.24e-2) =	2.2558e-1 (2.00e-2) =	2.7724e-1 (2.37e-2) -	2.1792e-1 (1.69e-2)
	70	<b>2.5197e-1 (3.22e-2)</b> =	2.5345e-1 (2.68e-2) =	2.9038e-1 (3.32e-2) -	3.3517e-1 (2.01e-2) -	2.5505e-1 (3.08e-2)
	100	3.0023e-1 (3.97e-2) =	3.0801e-1 (3.47e-2) =	3.0933e-1 (3.93e-2) =	3.1199e-1 (2.54e-2) =	<b>2.8910e-1 (2.05e-2)</b>
UF3	30	5.5194e-1 (2.17e-2) =	5.4897e-1 (2.52e-2) =	5.5622e-1 (2.24e-2) =	5.5530e-1 (1.79e-2) =	<b>5.4848e-1 (2.42e-2)</b>
	50	<b>5.2343e-1 (3.27e-2)</b> =	5.3470e-1 (4.53e-2) =	5.7054e-1 (5.51e-2) -	5.8262e-1 (3.28e-2) -	5.2924e-1 (3.81e-2)
	70	5.5801e-1 (4.80e-2) =	5.5178e-1 (7.38e-2) =	5.6621e-1 (5.97e-2) =	6.1754e-1 (6.48e-2) -	<b>5.5071e-1 (4.27e-2)</b>
	100	6.0984e-1 (4.71e-2) =	6.2062e-1 (3.99e-2) =	6.6248e-1 (3.57e-2) -	6.6933e-1 (6.29e-2) =	<b>6.0890e-1 (4.80e-2)</b>
UF4	30	1.4366e-1 (8.60e-3) =	1.5636e-1 (9.79e-3) -	1.5329e-1 (5.80e-3) -	1.4435e-1 (5.40e-3) -	<b>1.4198e-1 (8.38e-3)</b>
	50	1.7116e-1 (7.50e-3) =	1.7282e-1 (6.45e-3) =	1.7471e-1 (6.25e-3) =	1.7714e-1 (5.54e-3) =	<b>1.7111e-1 (6.38e-3)</b>
	70	1.9216e-1 (5.67e-3) -	1.8340e-1 (3.77e-3) -	1.8413e-1 (4.22e-3) -	1.9054e-1 (4.60e-3) -	<b>1.8202e-1 (4.82e-3)</b>
	100	1.9594e-1 (3.09e-3) =	2.3303e-1 (2.64e-3) -	2.3904e-1 (2.98e-3) -	2.4477e-1 (1.59e-3) -	<b>1.9259e-1 (2.44e-3)</b>
UF5	30	4.4486e+0 (4.60e-1) -	4.4632e+0 (5.25e-1) -	4.1110e+0 (4.54e-1) =	4.0750e+0 (4.86e-1) =	<b>4.0302e+0 (4.15e-1)</b>
	50	4.6777e+0 (4.73e-1) =	4.9744e+0 (3.91e-1) -	5.1982e+0 (4.58e-1) -	5.5532e+0 (3.85e-1) -	<b>4.6321e+0 (4.68e-1)</b>
	70	5.4601e+0 (5.03e-1) -	5.4031e+0 (4.25e-1) -	5.7797e+0 (3.88e-1) -	5.9045e+0 (3.82e-1) -	<b>4.9570e+0 (3.30e-1)</b>
	100	5.3708e+0 (3.57e-1) =	5.5233e+0 (3.10e-1) =	5.9807e+0 (3.16e-1) =	6.3421e+0 (2.80e-1) =	<b>5.3280e+0 (3.03e-1)</b>
UF6	30	3.1769e+0 (5.38e-1) +	3.4567e+0 (5.59e-1) =	4.3717e+0 (8.10e-1) =	<b>3.0582e+0 (7.32e-1)</b> +	3.4377e+0 (9.19e-1)
	50	4.1531e+0 (6.55e-1) =	<b>4.0647e+0 (5.56e-1)</b> =	4.1452e+0 (8.45e-1) =	4.2225e+0 (5.59e-1) =	4.1321e+0 (7.49e-1)
	70	4.4266e+0 (6.52e-1) =	<b>4.3203e+0 (6.17e-1)</b> =	4.6088e+0 (7.59e-1) =	4.8408e+0 (7.95e-1) =	4.6443e+0 (6.85e-1)
	100	<b>5.0072e+0 (3.32e-1)</b> +	5.4290e+0 (5.06e-1) =	5.7759e+0 (6.55e-1) =	6.0638e+0 (4.35e-1) =	5.3640e+0 (4.54e-1)
UF7	30	8.8024e-1 (1.24e-1) =	9.4785e-1 (1.50e-1) -	8.8933e-1 (1.51e-1) -	8.6867e-1 (1.41e-1) =	<b>8.6801e-1 (1.46e-1)</b>
	50	1.1872e+0 (9.32e-2) =	<b>1.1092e+0 (7.28e-2)</b> =	1.6355e+0 (1.33e-1) -	1.8907e+0 (1.18e-1) =	1.1737e+0 (9.11e-2)
	70	1.5201e+0 (1.07e-1) -	1.2965e+0 (1.33e-1) =	1.2854e+0 (1.55e-1) =	1.3466e+0 (1.07e-1) =	<b>1.2565e+0 (1.51e-1)</b>
	100	1.3446e+0 (9.91e-2) =	1.5784e+0 (7.24e-2) =	1.7716e+0 (1.30e-1) =	1.9868e+0 (9.51e-2) =	<b>1.3223e+0 (8.96e-2)</b>
UF8	30	3.6810e-1 (4.03e-2) =	3.7192e-1 (3.55e-2) =	3.6776e-1 (4.11e-2) =	3.6698e-1 (1.85e-2) =	<b>3.6567e-1 (3.91e-2)</b>
	50	4.4309e-1 (3.55e-2) =	4.5028e-1 (4.14e-2) =	4.8369e-1 (3.46e-2) =	4.8719e-1 (4.81e-2) =	<b>4.4216e-1 (5.41e-2)</b>
	70	4.7365e-1 (2.73e-2) =	4.5295e-1 (3.57e-2) =	4.5716e-1 (3.41e-2) =	4.8892e-1 (3.62e-2) =	<b>4.5163e-1 (2.30e-2)</b>
	100	<b>5.5156e-1 (4.33e-2)</b> =	5.6017e-1 (3.22e-2) =	5.9836e-1 (4.01e-2) =	6.2829e-1 (3.81e-2) =	5.5226e-1 (5.38e-2)
UF9	30	8.5075e-1 (3.14e-2) =	8.5169e-1 (6.82e-2) =	8.5059e-1 (2.35e-2) =	8.4404e-1 (3.88e-2) =	<b>8.4367e-1 (2.16e-2)</b>
	50	8.7778e-1 (8.41e-3) =	8.8007e-1 (1.34e-2) =	8.8907e-1 (1.37e-2) =	8.9675e-1 (1.10e-2) =	<b>8.7750e-1 (1.30e-2)</b>
	70	<b>8.6895e-1 (1.25e-2)</b> =	8.7388e-1 (1.81e-2) =	8.9791e-1 (1.57e-2) =	9.2164e-1 (1.49e-2) =	8.7486e-1 (1.57e-2)
	100	8.8605e-1 (1.88e-2) =	8.9036e-1 (2.34e-2) =	8.9476e-1 (1.85e-2) =	9.2333e-1 (1.41e-2) =	<b>8.8572e-1 (1.48e-2)</b>
MaF1	30	9.7104e-2 (2.46e-2) =	9.5869e-2 (1.68e-2) =	9.5800e-2 (2.06e-2) =	9.5290e-1 (2.31e-2) =	<b>9.4963e-2 (1.94e-2)</b>
	50	1.9888e-1 (5.62e-2) =	1.7558e-1 (6.45e-2) =	<b>1.7232e-1 (7.66e-2)</b> +	2.1007e-1 (7.36e-2) =	2.0126e-1 (6.19e-2)
	70	3.9300e-1 (9.46e-2) =	<b>3.7773e-1 (1.23e-1)</b> +	3.8632e-1 (1.31e-1) =	4.1021e-1 (1.64e-1) =	3.9776e-1 (1.21e-1)
	100	9.1686e-1 (2.65e-1) =	9.4291e-1 (2.66e-1) =	1.0050e+0 (2.60e-1) =	1.1362e+0 (1.95e-1) =	<b>9.0010e-1 (2.46e-1)</b>
MaF2	30	7.8712e-2 (2.15e-2) =	7.7855e-2 (1.48e-2) =	7.8580e-2 (8.17e-3) =	7.9049e-2 (1.32e-2) =	<b>7.6100e-2 (1.92e-2)</b>
	50	<b>8.9318e-2 (1.73e-2)</b> =	9.3992e-2 (1.52e-2) =	9.8968e-2 (1.38e-2) =	9.0956e-2 (1.83e-2) +	9.9196e-2 (1.28e-2)
	70	<b>1.0134e-1 (2.36e-2)</b> =	1.2068e-1 (1.64e-2) =	1.2153e-1 (1.63e-2) =	1.2261e-1 (1.87e-2) =	1.1752e-1 (1.91e-2)
	100	2.3762e-1 (2.65e-2) =	2.8609e-1 (2.44e-2) =	2.9525e-1 (2.56e-2) =	2.9162e-1 (2.10e-2) =	<b>1.8808e-1 (2.81e-2)</b>
MaF3	30	3.5450e+6 (8.46e+5) =	3.4622e+6 (6.12e+5) =	3.6572e+6 (5.88e+5) =	3.4773e+6 (6.23e+5) =	<b>3.3657e+6 (3.67e+5)</b>
	50	1.2357e+7 (2.14e+6) =	1.2553e+7 (1.48e+6) =	<b>1.2256e+7 (1.62e+6)</b> =	1.2715e+7 (1.03e+6) =	1.2841e+7 (1.37e+6)
	70	2.9380e+7 (2.72e+6) =	2.9507e+7 (1.85e+6) =	3.7639e+7 (3.32e+6) =	3.9171e+7 (2.40e+6) =	<b>2.9315e+7 (1.92e+6)</b>
	100	7.5861e+7 (4.38e+6) =	7.6695e+7 (2.03e+6) =	8.3108e+7 (3.02e+6) =	8.8108e+7 (3.91e+6) =	<b>6.5479e+7 (4.96e+6)</b>
MaF4	30	5.4854e+3 (9.36e+2) =	<b>5.4701e+3 (6.55e+2)</b> =	5.5005e+3 (7.89e+2) =	5.5197e+3 (3.70e+2) =	5.5648e+3 (8.45e+2)
	50	<b>1.0573e+4 (5.75e+2)</b> =	1.2625e+4 (8.43e+2) -	1.4800e+4 (8.65e+2) -	1.5177e+4 (6.20e+2) -	1.0708e+4 (6.29e+2)
	70	1.9902e+4 (6.91e+2) =	1.9492e+4 (8.62e+2) =	2.0538e+4 (1.32e+3) =	2.6328e+4 (9.34e+2) =	<b>1.6295e+4 (9.27e+2)</b>
	100	2.6804e+4 (1.16e+3) =	2.7681e+4 (1.29e+3) =	2.8050e+4 (1.33e+3) =	2.9401e+4 (1.33e+3) =	<b>2.4665e+4 (1.66e+3)</b>
MaF5	30	3.2664e+0 (8.28e-1) =	3.1495e+0 (1.17e+0) =	3.1242e+0 (1.33e+0) =	<b>3.1012e+0 (8.51e-1)</b> =	3.2530e+0 (7.86e-1)
	50	<b>5.0050e+0 (9.13e-1)</b> =	5.3148e+0 (8.96e-1) =	5.7126e+0 (7.91e-1) =	5.9004e+0 (1.44e+0) =	5.1353e+0 (1.11e+0)
	70	7.3467e+0 (1.10e+0) =	7.4882e+0 (1.69e+0) =	7.7235e+0 (1.28e+0) =	7.9052e+0 (1.60e+0) =	<b>7.3261e+0 (8.73e-1)</b>
	100	9.9119e+0 (1.47e+0) =	1.0305e+1 (1.46e+0) =	1.0119e+1 (1.04e+0) =	9.8280e+0 (1.57e+0) =	<b>9.6278e+0 (1.18e+0)</b>
MaF6	30	<b>7.8508e+0 (1.73e+0)</b> =	8.3266e+0 (1.90e+0) =	8.3194e+0 (4.61e+0) =	7.9752e+0 (4.69e+0) =	8.7198e+0 (4.57e+0)
	50	3.6619e+1 (1.86e+1) =	<b>3.3735e+1 (1.73e+1)</b> =	3.8843e+1 (1.38e+1) =	4.4109e+1 (1.43e+1) =	3.8636e+1 (1.32e+1)
	70	1.1294e+2 (4.32e+1) =	<b>1.0133e+2 (2.39e+1)</b> =	1.4794e+1 (2.15e+1) =	1.8929e+2 (1.70e+1) =	1.0348e+2 (2.66e+1)
	100	<b>1.7182e+2 (3.17e+1)</b> +	1.8930e+2 (3.39e+1) =	1.9126e+2 (5.15e+1) =	2.2385e+2 (4.10e+1) =	2.1881e+2 (2.93e+1)
MaF7	30	1.3145e+0 (2.17e-1) -	1.5436e+0 (3.48e-1) -	1.4108e+0 (3.42e-1) -	1.4702e+0 (3.72e-1) -	<b>8.5116e-1 (3.68e-1)</b>
	50	1.6399e+0 (6.64e-1) =	2.3934e+0 (6.92e-1) -	2.4815e+0 (7.67e-1) -	2.5443e+0 (6.14e-1) -	<b>1.6273e+0 (7.28e-1)</b>
	70	4.2512e+0 (1.26e+0) =	3.3526e+0 (9.96e-1) -	3.4353e+0 (8.44e-1) -	4.4556e+0 (6.32e-1) -	<b>1.6543e+0 (1.03e+0)</b>
	100	5.1679e+0 (6.90e-1) =	5.1070e+0 (5.21e-1) =	5.7788e+0 (5.35e-1) =	5.6507e+0 (1.13e+0) =	<b>3.8288e+0 (7.07e-1)</b>
+/-/=		5/8/51	4/16/44	4/28/32	1/32/31	

Table A.5: Comparisons of SAEA-SHT with Five Algorithms on the UF and MaF Test Suites

Problem	D	SAEA-SHT-I	ExI	LCB	PoI	SAEA-SHT-E	SAEA-SHT-U	SAEA-SHT
UF1	30	9.39e-1 (1.38e-1) -	8.13e-1 (1.08e-1) =	1.18e+0 (1.16e-1) -	8.55e-1 (1.85e-1) =	1.15e+0 (8.73e-2) -	1.04e+0 (1.63e-1) -	<b>7.52e-1 (1.04e-1)</b>
	50	1.21e+0 (1.12e-1) -	1.21e+0 (1.52e-1) -	1.00e+0 (8.76e-2) =	1.34e+0 (1.24e-1) -	1.32e+0 (1.21e-1) -	1.33e+0 (1.08e-1) -	<b>1.06e+0 (1.17e-1)</b>
	70	1.36e+0 (1.93e-1) -	1.36e+0 (1.33e-1) -	1.13e+0 (7.78e-2) =	1.49e+0 (7.98e-2) -	1.45e+0 (7.15e-2) -	1.48e+0 (7.61e-2) -	<b>1.15e+0 (1.30e-1)</b>
	100	1.51e+0 (8.05e-2) -	1.51e+0 (8.25e-2) -	1.49e+0 (5.88e-2) =	1.62e+0 (5.92e-2) -	1.54e+0 (6.08e-2) -	1.58e+0 (6.92e-2) -	<b>1.30e+0 (1.33e-1)</b>
UF2	30	<b>1.50e-1 (2.11e-2) +</b>	1.78e-1 (2.45e-2) =	4.03e-1 (1.19e-1) -	1.75e-1 (1.31e-2) =	2.37e-1 (4.11e-2) -	1.52e-1 (1.22e-2) +	1.74e-1 (1.61e-2)
	50	1.77e-1 (1.22e-1) +	5.60e-1 (1.03e-1) -	6.71e-1 (3.08e-2) -	5.92e-1 (1.23e-1) -	4.02e-1 (3.86e-2) -	<b>1.73e-1 (1.10e-2) +</b>	2.17e-1 (1.69e-2)
	70	<b>2.15e-1 (5.35e-2) +</b>	6.34e-1 (5.75e-2) -	7.04e-1 (3.05e-2) -	6.99e-1 (2.77e-2) -	4.13e-1 (7.09e-2) -	2.20e-1 (2.87e-2) +	2.55e-1 (3.08e-2)
	100	<b>2.55e-1 (4.10e-2) +</b>	6.65e-1 (4.42e-2) -	7.45e-1 (1.80e-2) -	7.39e-1 (1.98e-2) -	4.68e-1 (2.98e-2) -	2.66e-1 (1.73e-2) +	2.89e-1 (2.05e-2)
UF3	30	5.76e-1 (1.10e-1) =	5.76e-1 (1.52e-1) =	9.62e-1 (1.39e-1) -	5.62e-1 (2.88e-2) =	<b>5.20e-1 (2.52e-2) +</b>	6.25e-1 (1.75e-1) =	5.48e-1 (2.42e-2)
	50	6.50e-1 (1.89e-1) -	8.50e-1 (1.57e-1) -	9.92e-1 (6.39e-2) -	9.81e-1 (4.95e-2) -	<b>4.94e-1 (1.77e-2) +</b>	6.54e-1 (1.45e-1) =	5.29e-1 (3.81e-2)
	70	1.92e-1 (2.43e-3) -	8.88e-1 (8.54e-2) -	1.00e+0 (3.29e-2) -	9.45e-1 (4.36e-2) -	5.37e-1 (4.93e-2) -	5.86e-1 (5.31e-2) =	5.50e-1 (4.27e-2)
	100	<b>6.55e-1 (6.00e-2) +</b>	9.09e-1 (5.90e-2) -	9.79e-1 (3.61e-2) -	9.66e-1 (4.75e-2) -	6.87e-1 (4.85e-2) +	6.45e-1 (5.23e-2) =	6.08e-1 (4.80e-2)
UF4	30	1.57e-1 (9.33e-3) -	1.57e-1 (9.03e-3) -	1.43e-1 (1.15e-2) =	1.61e-1 (7.01e-3) -	1.46e-1 (5.28e-3) =	1.73e-1 (4.51e-3) -	<b>1.41e-1 (8.38e-3)</b>
	50	1.87e-1 (2.91e-3) -	1.87e-1 (2.75e-3) -	1.73e-1 (3.64e-3) =	1.89e-1 (4.69e-3) -	1.76e-1 (1.01e-3) =	1.83e-1 (7.59e-3) -	<b>1.71e-1 (6.38e-3)</b>
	70	1.92e-1 (2.44e-3) -	1.92e-1 (2.37e-3) -	1.83e-1 (3.55e-3) =	1.95e-1 (1.74e-3) =	1.83e-1 (1.51e-3) =	1.91e-1 (4.67e-3) -	<b>1.82e-1 (4.82e-3)</b>
	100	1.96e-1 (1.76e-3) -	1.96e-1 (1.96e-3) -	1.93e-1 (2.35e-3) =	1.99e-1 (1.80e-3) -	2.14e-1 (1.66e-3) =	1.95e-1 (2.84e-3) -	<b>1.92e-1 (2.44e-3)</b>
UF5	30	3.62e+0 (5.01e-1) +	<b>3.60e+0 (5.85e-1) +</b>	4.82e+0 (2.84e-1) -	4.42e+0 (3.75e-1) -	4.15e+0 (1.63e-1) =	4.09e+0 (3.05e-1) =	4.03e+0 (4.15e-1)
	50	4.76e+0 (3.87e-1) =	4.86e+0 (4.07e-1) =	5.47e+0 (1.08e-1) -	5.36e+0 (1.56e-1) -	4.42e+0 (1.06e-1) =	4.66e+0 (3.24e-1) =	<b>4.63e+0 (4.68e-1)</b>
	70	5.03e+0 (1.28e-1) =	5.17e+0 (1.58e-1) =	5.75e+0 (1.87e-1) -	5.72e+0 (1.66e-1) -	4.88e+0 (1.90e-1) =	4.82e+0 (2.42e-1) =	<b>4.95e+0 (3.30e-1)</b>
	100	<b>5.01e+0 (1.85e-1) +</b>	5.04e+0 (2.06e-1) +	5.87e+0 (2.16e-1) -	6.06e+0 (1.55e-1) -	5.45e+0 (1.22e-1) =	5.46e+0 (1.52e-1) =	5.32e+0 (3.03e-1)
UF6	30	3.70e+0 (8.01e-1) =	3.70e+0 (8.18e-1) =	3.48e+0 (5.98e-1) =	4.14e+0 (7.30e-1) =	<b>3.01e+0 (5.39e-1) +</b>	4.82e+0 (5.51e-1) -	3.43e+0 (9.19e-1)
	50	5.40e+0 (5.23e-1) -	5.40e+0 (5.57e-1) -	4.16e+0 (4.16e-1) =	4.15e+0 (5.75e-1) =	5.41e+0 (2.44e-1) =	6.02e+0 (4.20e-1) =	<b>4.13e+0 (7.49e-1)</b>
	70	5.46e+0 (4.38e-1) -	5.46e+0 (4.08e-1) -	4.43e+0 (4.31e-1) =	4.66e+0 (3.49e-1) =	5.89e+0 (2.54e-1) -	6.17e+0 (2.52e-1) -	<b>4.64e+0 (6.85e-1)</b>
	100	6.19e+0 (3.17e-1) -	6.19e+0 (3.71e-1) -	5.33e+0 (3.28e-1) =	5.30e+0 (3.59e-1) =	<b>5.02e+0 (2.42e-1) +</b>	6.41e+0 (3.32e-1) -	5.36e+0 (4.54e-1)
UF7	30	8.94e-1 (1.26e-1) -	8.68e-1 (1.63e-1) =	1.23e+0 (9.31e-2) -	1.02e+0 (2.27e-1) =	1.25e+0 (5.03e-2) -	1.06e+0 (2.42e-1) -	<b>8.68e-1 (1.46e-1)</b>
	50	1.94e+0 (9.37e-2) -	1.24e+0 (9.07e-2) =	1.42e+0 (7.45e-2) -	1.42e+0 (1.08e-1) -	1.33e+0 (8.91e-2) -	1.36e+0 (1.24e-1) -	<b>1.17e+0 (9.11e-2)</b>
	70	1.83e+0 (9.29e-2) -	1.33e+0 (9.15e-2) =	1.57e+0 (6.21e-2) -	1.56e+0 (5.85e-2) -	1.45e+0 (6.71e-2) -	1.54e+0 (6.08e-2) -	<b>1.25e+0 (1.51e-1)</b>
	100	1.50e+0 (9.26e-2) -	1.50e+0 (9.44e-2) -	1.65e+0 (6.27e-2) -	1.71e+0 (4.45e-2) -	1.58e+0 (2.51e-2) -	1.63e+0 (7.50e-2) -	<b>1.32e+0 (8.96e-2)</b>
UF8	30	<b>3.30e-1 (4.10e-2) +</b>	3.66e-1 (4.43e-2) =	1.06e+0 (2.72e-1) -	3.94e-1 (2.25e-2) =	3.31e-1 (6.18e-2) +	9.65e-1 (4.43e-1) -	3.65e-1 (3.91e-2)
	50	<b>4.05e-1 (6.20e-2) +</b>	2.35e+0 (6.85e-1) -	3.11e+0 (1.51e-1) -	3.00e+0 (5.02e-1) -	4.19e-1 (1.67e-2) +	3.15e+0 (2.71e-1) -	4.42e-1 (5.41e-2)
	70	<b>4.21e-1 (3.76e-2) +</b>	2.91e+0 (3.49e-1) -	3.45e+0 (1.82e-1) -	3.34e+0 (1.53e-1) -	4.26e-1 (2.10e-2) +	3.28e+0 (1.53e-1) -	4.51e-1 (2.30e-2)
	100	5.11e-1 (3.45e-2) +	3.13e+0 (3.06e-1) -	3.48e+0 (1.69e-1) -	3.58e+0 (1.25e-1) -	<b>5.09e-0 (3.46e-2) +</b>	3.51e+0 (1.59e-1) -	5.52e-1 (5.38e-2)
UF9	30	8.88e-1 (4.20e-2) -	8.43e-1 (4.17e-2) =	1.13e+0 (2.19e-1) -	8.46e-1 (6.31e-2) =	1.07e+0 (1.42e-1) -	1.31e+0 (8.60e-1) -	<b>8.43e-1 (2.16e-2)</b>
	50	2.50e+0 (7.33e-1) -	2.50e+0 (7.71e-1) -	8.99e-0 (4.31e-1) =	2.99e+0 (7.18e-1) -	1.23e+0 (2.63e-1) -	3.25e+0 (2.29e-1) -	<b>8.77e-1 (1.30e-2)</b>
	70	2.91e+0 (3.10e-1) -	2.91e+0 (3.08e-1) -	3.49e+0 (1.65e-1) =	3.37e+0 (1.80e-1) -	1.68e+0 (1.50e-1) -	3.33e+0 (2.39e-1) -	<b>8.74e-1 (1.57e-2)</b>
	100	3.18e+0 (2.22e-1) -	3.18e+0 (2.72e-1) -	3.64e+0 (1.15e-1) =	3.66e+0 (1.22e-1) -	2.03e+0 (2.36e-1) -	3.73e+0 (9.14e-2) -	<b>8.85e-1 (1.48e-2)</b>
MaF1	30	1.45e-1 (1.20e-2) -	1.45e-1 (1.90e-2) -	6.87e-1 (5.35e-2) -	2.31e-1 (2.92e-2) =	4.01e-1 (3.23e-2) =	9.61e-2 (2.89e-2) =	<b>9.49e-2 (1.94e-2)</b>
	50	2.32e+0 (3.29e-1) -	2.32e+0 (3.53e-1) -	2.82e+0 (5.03e-1) -	2.71e+0 (3.63e-1) -	9.75e-1 (1.53e-1) -	2.00e-1 (2.65e-2) =	<b>2.01e-1 (6.19e-2)</b>
	70	3.43e+0 (4.86e-1) -	3.43e+0 (4.87e-1) -	3.60e+0 (1.17e+0) -	4.74e+0 (3.28e-1) -	2.04e+0 (2.53e-1) -	4.46e-1 (3.20e-1) =	<b>3.97e-1 (1.21e-1)</b>
	100	5.95e+0 (7.02e-1) -	5.95e+0 (7.35e-1) -	6.93e+0 (5.27e-1) -	7.02e+0 (2.95e-1) -	3.04e+0 (1.53e-1) -	8.91e-1 (2.83e-1) =	<b>9.00e-1 (2.46e-1)</b>
MaF2	30	8.29e-2 (5.63e-3) -	7.75e-2 (6.77e-3) =	1.14e-1 (1.43e-2) -	9.63e-2 (6.47e-3) -	8.86e-2 (2.03e-3) =	1.41e-1 (3.41e-3) -	<b>7.61e-2 (1.92e-2)</b>
	50	2.15e-1 (1.33e-2) -	2.15e-1 (1.13e-2) -	2.37e-1 (6.05e-3) -	2.33e-1 (8.50e-3) -	1.36e-1 (7.25e-3) -	2.37e-1 (8.27e-3) -	<b>9.91e-2 (1.28e-2)</b>
	70	3.13e-1 (1.47e-2) -	3.13e-1 (1.51e-2) -	3.43e-1 (1.20e-2) -	3.46e-1 (9.13e-3) -	1.26e-1 (3.80e-2) =	3.45e-1 (7.96e-3) -	<b>1.17e-1 (1.91e-2)</b>
	100	4.86e-1 (1.09e-2) -	4.86e-1 (1.89e-2) -	5.07e-1 (1.89e-2) -	5.05e-1 (1.66e-2) -	2.87e-1 (8.58e-4) -	5.05e-1 (1.76e-2) -	<b>1.88e-1 (2.81e-2)</b>
MaF3	30	3.52e+6 (7.28e+5) =	3.52e+6 (7.17e+5) =	4.31e+6 (1.15e+6) -	5.90e+6 (1.42e+6) -	4.76e+6 (7.34e+5) =	3.48e+6 (5.37e+5) =	<b>3.36e+6 (3.67e+5)</b>
	50	1.76e+7 (3.36e+6) -	1.76e+7 (3.39e+6) -	2.13e+7 (3.04e+6) -	2.27e+7 (2.44e+6) -	1.72e+7 (1.51e+6) -	2.20e+7 (2.99e+6) -	<b>1.28e+7 (1.37e+6)</b>
	70	4.01e+7 (7.56e+6) -	4.01e+7 (7.89e+6) -	3.98e+7 (6.25e+6) -	4.42e+7 (5.60e+6) -	3.64e+7 (4.24e+6) -	<b>2.25e+7 (3.43e+6) +</b>	2.93e+7 (1.92e+6)
	100	7.92e+7 (1.98e+7) -	7.92e+7 (1.65e+7) -	9.38e+7 (1.02e+7) -	9.44e+7 (7.06e+6) -	7.55e+7 (5.17e+6) -	9.58e+7 (8.97e+6) -	<b>6.54e+7 (4.96e+6)</b>
MaF4	30	4.94e+3 (6.03e+2) +	5.00e+3 (6.76e+2) +	<b>4.41e+3 (9.22e+2) +</b>	6.17e+3 (5.64e+2) =	6.36e+3 (5.80e+2) =	6.54e+3 (2.45e+2) =	5.56e+3 (8.45e+2)
	50	1.10e+4 (1.27e+3) =	1.09e+4 (1.10e+3) =	1.08e+4 (9.09e+2) =	1.11e+4 (7.14e+2) =	1.18e+4 (6.64e+2) =	1.14e+4 (7.53e+2) =	<b>1.07e+4 (6.29e+2)</b>
	70	1.20e+4 (2.07e+3) +	<b>1.14e+4 (1.57e+3) +</b>	1.69e+4 (9.87e+2) =	1.70e+4 (8.03e+2) =	1.71e+4 (7.51e+2) =	1.77e+4 (7.98e+2) =	1.62e+4 (9.27e+2)
	100	2.55e+4 (1.33e+3) =	2.49e+4 (1.60e+3) =	2.52e+4 (1.68e+3) =	2.55e+4 (6.18e+2) =	2.64e+4 (3.73e+2) -	2.56e+4 (1.20e+3) =	<b>2.46e+4 (1.66e+3)</b>
MaF5	30	3.72e+0 (7.29e-1) =	3.72e+0 (7.37e-1) =	5.93e+0 (8.74e-1) -	4.39e+0 (1.19e+0) -	6.24e+0 (3.92e-1) -	3.83e+0 (9.75e-1) =	<b>3.25e+0 (7.86e-1)</b>
	50	8.10e+0 (1.87e+0) =	8.10e+0 (1.09e+0) =	9.58e+0 (8.70e-1) -	9.05e+0 (1.08e+0) =	7.99e+0 (1.90e+0) =	9.35e+0 (1.55e+0) =	<b>5.13e+0 (1.11e+0)</b>
	70	1.13e+1 (1.79e+0) =	1.13e+1 (2.00e+0) =	1.27e+1 (9.36e-1) -	1.26e+1 (1.38e+0) =	1.14e+1 (1.80e+0) =	1.24e+1 (1.10e+0) =	<b>7.32e+0 (8.73e-1)</b>
	100	1.47e+1 (1.38e+0) -	1.47e+1 (1.14e+0) -	1.75e+1 (1.46e+0) -	1.78e+1 (1.30e+0) -	1.73e+1 (1.43e+0) -	1.69e+1 (6.33e-1) -	<b>9.62e+0 (1.18e+0)</b>
MaF6	30	1.25e+1 (4.43e+0) =	1.25e+1 (4.31e+0) =	4.24e+1 (1.43e+1) -	2.42e+1 (5.64e+0) -	2.40e+1 (7.26e+0) -	6.83e+1 (9.18e+0) -	<b>8.71e+0 (4.57e+0)</b>
	50	1.88e+2 (3.33e+1) -	1.88e+2 (3.34e+1) -	2.59e+2 (1.78e+1) -	2.41e+2 (2.56e+1) -	6.49e+1 (1.34e+1) -	2.37e+2 (1.69e+1) -	<b>3.86e+1 (1.32e+1)</b>
	70	3.13e+2 (2.48e+1) -	3.13e+2 (2.43e+1) -	3.63e+2 (3.14e+1) -	3.66e+2 (2.35e+1) -	1.66e+2 (6.02e+0) -	3.62e+2 (6.02e+0) -	<b>1.03e+2 (2.66e+1)</b>
	100	4.97e+2 (5.09e+1) -	4.97e+2 (5.68e+1) -	5.66e+2 (6.00e+1) -	5.74e+2 (3.26e+1) -	<b>1.88e+2 (4.46e+1) +</b>	5.59e+2 (5.59e+1) -	2.18e+2 (2.93e+1)
MaF7	30	9.92e-1 (3.48e-1) -	9.62e-1 (3.92e-1) =	1.23e+0 (2.18e+0) =	1.05e+0 (3.54e-1) =	7.10e+0 (9.69e-1) =	7.85e+0 (1.06e+0) =	<b>8.51e-1 (3.68e-1)</b>
	50	8.41e+0 (7.37e-1) -	8.41e+0 (7.89e-1) -	9.27e+0 (6.98e-1) =	8.36e+0 (1.32e+0) =	1.54e+0 (6.83e-1) =	9.14e+0 (5.97e-1) -	<b>1.62e+0 (7.28e-1)</b>
	70	8.91e+0 (8.87e-1) -	8.91e+0 (8.58e-1) -	9.50e+0 (4.59e-1) -	9.69e+0 (3.61e-1) -	1.58e+0 (5.62e-1) =	9.64e+0 (4.89e-1) -	<b>1.65e+0 (1.03e+0)</b>
	100	9.59e+0 (3.37e-1) -	9.59e+0 (3.70e-1) -	9.76e+0 (3.04e-1) -	9.80e+0 (3.15e-1) -	3.71e+0 (5.92e-1) =	1.00e+1 (4.46e-1) -	<b>3.82e+0 (0.70e-1)</b>
+/-=		14/42/8	4/43/17	1/45/18	0/49/15	11/38/15	5/44/15	

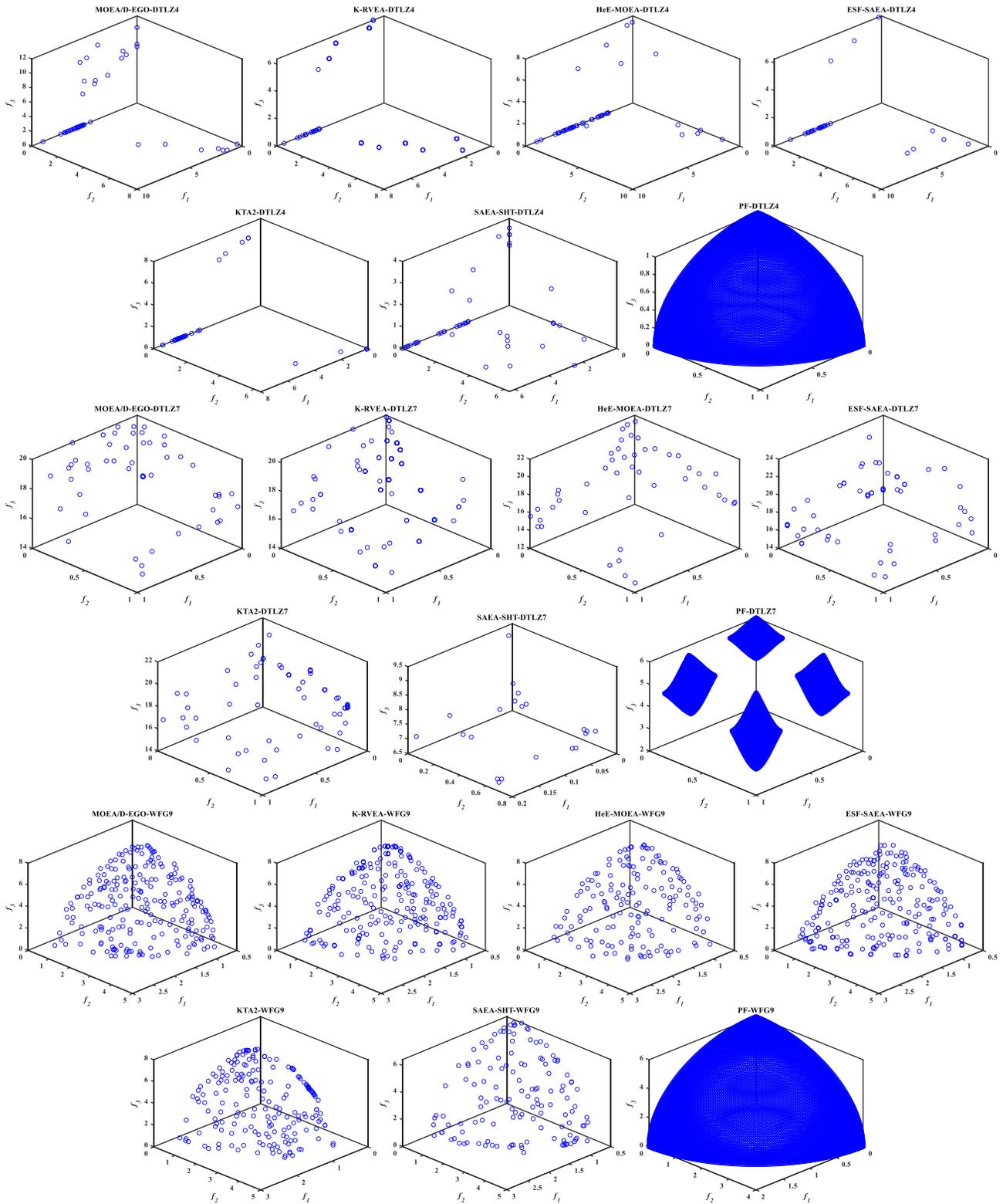


Figure A.1: The final populations obtained by all the compared algorithms on 70-D DTLZ4, DTLZ7, and WFG9

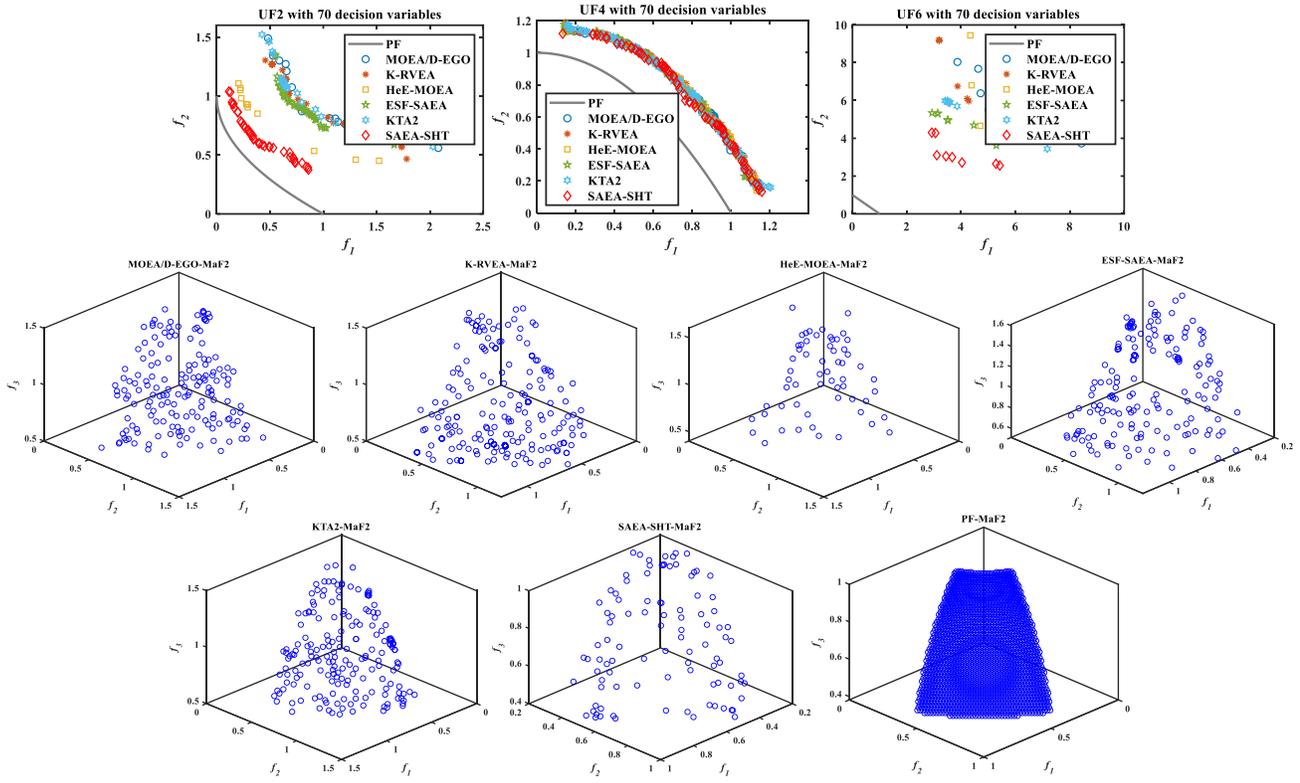


Figure A.2: The final populations obtained by all the compared algorithms on 70-D UF2, UF4, UF6, and MaF2