Information Sciences 569 (2021) 430–449

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An incremental-learning model-based multiobjective estimation of distribution algorithm

Tingrui Liu [a], Xin Li [b], Liguo Tan [c,*], Shenmin Song [a,*]

[a] Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin 150001, China
[b] Sino-German Robotics School, Shenzhen Institute of Information Technology, Shenzhen 518172, China
[c] Research Center of Basic Space Science, Harbin Institute of Technology, Harbin 150001, China

ARTICLE INFO

Article history:
Received 10 June 2020
Received in revised form 15 March 2021
Accepted 2 April 2021
Available online 8 April 2021

Keywords:
Evolutionary algorithm
Multiobjective optimization
Estimation of distribution
Incremental learning
Gaussian mixture model

ABSTRACT

Knowledge obtained from the properties of a Pareto-optimal set can guide an evolutionary search. Learning models for multiobjective estimation of distributions have led to improved search efficiency, but they incur a high computational cost owing to their use of a repetitive learning or iterative strategy. To overcome this drawback, we propose an algorithm for incremental-learning model-based multiobjective estimation of distributions. A learning mechanism based on an incremental Gaussian mixture model is embedded within the search procedure. In the proposed algorithm, all new solutions generated during the evolution are passed to a data stream, which is fed incrementally into the learning model to adaptively discover the structure of the Pareto-optimal set. The parameters of the model are updated continually as each newly generated datum is collected. Each datum is learned only once for the model, regardless of whether it has been preserved or deleted. Moreover, a sampling strategy based on the learned model is designed to balance the exploration/exploitation dilemma in the evolutionary search. The proposed algorithm is compared with six state-of-the-art algorithms for several benchmarks. The experimental results show that there is a significant improvement over the representative algorithms.

1. Introduction

In many engineering tasks, the outcome naturally has multiple expected properties. We have a multiobjective optimization problem (MOP) when we do not know how to balance the properties with a priori weight but must consider each of them as a separate objective function. In this case, we have to optimize simultaneously two or more objective functions that may conflict with each other. Thus, there may not be a unique optimal solution that achieves the best outcome for each objective. For MOPs, we aim to find a set of good compromises for the decision maker [1], which is called the Pareto-optimal set (PS). The objective vectors of the PS cover the Pareto front (PF).

An evolutionary algorithm, since its population-based nature perfectly matches the solution set required for MOPs, could be adopted to build a representative subset of the PS in a single run. Thus, the multiobjective evolutionary algorithm (MOEA) has become the preferred method for addressing complicated MOPs. Extensive studies have been carried out to design and improve MOEAs in the last three decades. One of the burgeoning trends is integrating a machine learning mechanism into the framework of MOEA [2], because using the knowledge learned from the population would enhance the efficiency of the

* Corresponding authors.
E-mail addresses: liutingrui@126.com (T. Liu), eshing_li@126.com (X. Li), tanliguo@hit.edu.cn (L. Tan), songshenmin@hit.edu.cn (S. Song).

https://doi.org/10.1016/j.ins.2021.04.011
0020-0255/

evolutionary search. The PS of a continuous MOP with m objectives is a piecewise continuous $(m-1)$-dimensional manifold under mild conditions. Since the manifold property of the PS was reported for RM-MEDA [3], many supervised and unsupervised learning methods, such as Gaussian processes, Bayesian networks, neural networks, restricted Boltzmann machines, and self-organizing maps (SOMs), have been applied to learn and estimate the PS of a MOP to produce high-quality offspring.

Work on learning-based MOEAs focuses on learning the property of the PS, mainly from the following two aspects. On the one hand, a learning model of the manifold structure can be used to approximate the PS directly. For example, local principal component analysis was applied to build a probability distribution for the manifold structure of a PS in RM-MEDA. Zhou et al. [4] proposed a MOEA based on decomposition and probabilistic modeling, in which a multivariate Gaussian model was established to extract information about the local and global distributions to generate offspring. Liu et al. [5] designed a Baldwinian learning operator to obtain the descent direction of an evolutionary search based on a distribution model of the current population. On the other hand, instead of approximating the PS with models, some researchers have designed a mating restriction mechanism based on the manifold property of the PS. These MOEAs divide the population into several subpopulations, in which individuals with high similarity and mutual recombination approximate the PS. Zhang et al. [6] defined neighborhood relationships among solutions by utilizing a SOM to establish the topological structure of the population. Li et al. [7] used the k-means clustering algorithm to extract information about the neighborhood relationship for each individual for the mating restriction. Sun et al. [8] designed an indicator based on information from subpopulations. It adaptively indexed the different requirement levels in the exploration/exploitation compromise.

However, the machine learning model in MOEA always has a high computational cost [9], which mainly arises from two factors. First, the iterative strategies used during learning inherently have a high computational cost [7]. The algorithms usually learn the knowledge from the entire population of solutions through multiple iterations until converging to a well-pleasing result. They appear to have little effect early in the evolutionary process, as the initial solutions differ widely from the theoretical optimal solutions [10]. Second, the high computational cost stems from repetitive learning [8]. Some solutions, which survive the fitness selection, are repeatedly used for model learning, despite being unable to provide new knowledge for the model. Repetitive learning does not modify the model, so adds to the computational cost.

To decrease the computational cost, a feasible approach is to have fewer iterations in learning. Zhang et al. [11] adopted a SOM for the entire population, with only one iteration for each generation. However, some solutions survive several environmental selections and are still repeatedly used for SOM learning. Liu et al. [12] proposed that the learning machine is not run unless the population of adjacent generations has no significant similarity. Cheng et al. [13] constructed Gaussian process-based inverse models from the objective space to the decision space to find nondominated solutions. A random grouping technique was used to reduce the number of inverse models. Li et al. [14] incorporated manifold learning into multiobjective optimization. This nonlinear dimensionality-reduction technique extracts the geometric properties of low-dimensional manifolds embedded in the high-dimensional ambient space. However, these approaches still have unnecessary computational costs due to the iterations or repetitive learning when the population approximates the PS.

Inspired by data stream learning [15], an incremental-learning model-based multiobjective algorithm to estimate distributions, named ILME, is proposed and investigated. In ILME, all new solutions generated in the evolutionary process are combined into a data stream, which is called the evolving data stream. Each new generated solution is a datum in the evolving data stream. The evolving data stream is fed into a learning mechanism based on an incremental Gaussian mixture model (IGMM), which is employed to discover the manifold structure of the PS. The parameters of the model are updated continually as newly generated data are collected. Each datum is learned only once by the incremental model, regardless of whether it has been preserved or deleted from the population. Moreover, the sampling strategy based on the learned model is designed to balance the exploration/exploitation dilemma in the evolutionary search.

This paper is organized as follows. Section 2 briefly discusses related work and the IGMM. Section 3 describes the algorithmic framework of ILME and its components. The numerical experiments are considered in Section 4. The algorithmic sensitivity of the parameters and further experiments are discussed in Section 5. Section 6 concludes the paper and suggests future research avenues.

2. Related work

2.1. Background

A box-constrained continuous MOP could be formulated mathematically as follows:

$$\begin{aligned} \min \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^{\mathrm{T}} \\ \text{s.t.} \quad & \mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}} \in \Omega \subset \mathbb{R}^n \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is a n-dimensional decision vector. Ω defines the feasible decision (search) space in the n-dimensional decision space $\mathbb{R}^n$. $\mathbf{F}: \Omega \to \mathbb{R}^m$ is a mapping from the decision space to the objective space, which consists of m objective functions. For two solutions $\mathbf{u}, \mathbf{v} \in \Omega, \mathbf{u}$ is said to dominate $\mathbf{v}$ if and only if $f_i(\mathbf{u}) \leqslant f_i(\mathbf{v})$ for all $i \in \{1, 2, \ldots, m\}$ and $\mathbf{F}(\mathbf{u}) \neq \mathbf{F}(\mathbf{v})$. A solution is Pareto optimal if no other solution in Ω dominates it. The set of all Pareto optimal solutions constitutive of Pareto optimal set (PS), while the objective vectors of PS is named Pareto front (PF), denoted by $\mathrm{PF} = \{\mathbf{F}(\mathbf{x}) \in \mathbb{R}^m | \mathbf{x} \in \mathrm{PS}\}$.

Many remarkable efforts have been devoted to designing and improving MOEAs for MOPs in the last three decades. Roughly, MOEAs can be based on dominance, a metric, or decomposition. Dominance-based MOEAs adopt the Pareto dominance relationship to maintain convergence preferentially. Other strategies are used to preserve diversity. Examples include NSGA-II [16], PESA-II [17], and NSGA-III [18]. Metric-based MOEAs employ a performance metric (e.g., *hypervolume*, R2, or Δ_p) as the optimization objective to guide the selection of solutions. Examples include SMS-EMOA [19], R2-IBEA [20], and DDE [21]. Decomposition-based MOEAs decompose the original MOPs into a set of single-objective optimization problems or simplified MOPs, and optimize them with a collaborative method. Examples include MOEA/D [22], TMOEA/D [23], and MOEA/D-LTD [2]. In addition, MOEAs are often applied to aid decision-making [1].

2.2. Multiobjective estimation of distribution algorithms

The proposed algorithm is type of a multiobjective estimation of distribution algorithm (MOEDA), which is a modified form of the estimation of distribution algorithm (EDA). The modeling method and the fitness assignment function are the two major components of a MOEDA. Its fitness assignment function usually comes from a MOEA, and most often in current research, a Pareto dominance-based approach is used. The modeling method adopted in MOEDA is based either on a mixture of distributions or on a graphical model.

MOEDAs based on graphical models lean mostly upon Bayesian network, which is a graphical expression of the conditional relationships among the stochastic variables. One of the most widely used algorithms is the multiobjective Bayesian optimization algorithm [24], which exploits the fitness assignment of NSGA-II. A hierarchical Bayesian optimization algorithm [25] uses a strength criterion from SPEA, but combines *k*-means in the objective space to improve scalability. By estimating a Gaussian Bayesian network to cluster the solutions, the diversity-preserving multiobjective real-coded Bayesian optimization algorithm [26] transforms the problem variables and utilizes a diversity-preserving selection mechanism, which uses a dynamic crowding and adaptive sharing approach. Martins et al. [27] developed a hybrid multiobjective Bayesian estimation of distribution algorithm (HMOBEDA), which scrutinizes a probabilistic graphic model based on a Bayesian network. The probabilistic graphic model gives the joint probability of decisions, objectives, variables, and configuration parameters to assess the influence of both diversity and convergence along the approximate PF. Martins et al. [28] improved HMOBEDA using the information in the final structure of the probabilistic graphic model. Garrido et al. [29] designed an information-based predictive entropy search for the simultaneous optimization of multiple expensive-to-evaluate black-box functions under the presence of several constraints.

MOEDAs based on a mixture of distributions explicitly employ a mixture of probability distributions to directly approximate the PS. Bosman et al. [30] proposed multiobjective iterated density estimation algorithms (MIDEAs) as a paradigm for MOEDA, in which a probabilistic model is learned to stimulate the desirable parallel exploration along the PF. Li et al. [31] proposed a hybrid EDA with a joint probability distribution for multiobjective 0/1 knapsack problems. The adaptive variance scaling was enhanced by introducing a standard deviation ratio trigger embedded within the normal mixture distribution in MIDEA [32]. This approach avoids the loss of diversity and premature convergence. Further, adaptive variance scaling has been conjointly employed with an anticipated mean shift in a multiobjective adapted maximum-likelihood Gaussian mixture model (GMM) [33]. Shim et al. [34] introduced a restricted Boltzmann machine, which is an energy-based stochastic neural network, into MOEDA. The energy function of the network is used to construct a probabilistic model. Mohagheghi et al. [35] adopted a Voronoi mesh to construct the stochastic model. Maza et al. [36] integrated the mutual information between random variables as a probabilistic model to guide the search by modeling the redundancy and relevance relations between features in intrusion detection systems.

Apart from the above-mentioned research, "learnable" MOEDA represents a new frontier in research on MOEDA and is a paradigm that integrates MOEDA solvers with knowledge learning to achieve better optimization efficiency and performance. Based on this paradigm, MOEDAs with different learning algorithms, such as manifold learning [37], generative adversarial networks [38], and growing neural gas [9], have been developed. For example, SMEA [11] is a newly proposed competitive MOEA, which learns a self-organizing mapping in decision space to model the PF for assisting the evolutionary search for the PS. MDESL [39] carries out a *k*-means clustering analysis on the current population to extract knowledge about neighborhood relationships for mating restriction. MOEA/D-LTD [2] uses a Gaussian process-based model to learn the mapping from $(m-1)$ objective functions to the remaining objective function (m is the number of objectives) for characterizing the estimated PF. As mentioned earlier, these MOEDAs suffer from the problem of high computational cost.

2.3. Incremental Gaussian mixture model

IGMMs [15], concept formation algorithms based on unsupervised learning, aim to learn a GMM incrementally from the data stream. IGMMs have excellent modeling ability and can form smooth approximations for arbitrarily shaped densities. They can accommodate components of different sizes and correlation structures. Compared with batch learning, incremental learning does not usually require a significant amount of computing resources. Hence, we use an IGMM in incremental learning for the generated solutions during the evolutionary process. A GMM with K components in a D-dimensional space can be defined as follows:

$$\begin{cases} p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\} \\ \sum_{k=1}^{K} \pi_k = 1, \ \pi_k \geqslant 0 \end{cases} \tag{2}$$

where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ is the set of model parameters and π_k is the prior of the k-th Gaussian component with mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$.

Algorithm 1. Incremental Gaussian Mixture Model

Require: the existent model parameters $\Theta_0 = \{\pi_k^0, \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0\}_{k=1}^{K}$, the number of collected data N, the newly available data $\{\mathbf{x}^*\}$.

Ensure: the newly model parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$.

1: **while** $|L(\Theta_{i+1}) - L(\Theta_i)| \leqslant \Delta$ **do**

2: E-step: Evaluate the posterior probabilities, i.e. the membership weight of the newly available data $\mathbf{x}^*$ in mixture component K, by:

$$p(k|\mathbf{x}^*) = \frac{\pi_k^i \mathcal{N}(\mathbf{x}^*|\boldsymbol{\mu}_k^i, \boldsymbol{\Sigma}_k^i)}{\sum_{j=1}^{K} \pi_j^i \mathcal{N}(\mathbf{x}^*|\boldsymbol{\mu}_j^i, \boldsymbol{\Sigma}_j^i)}, \quad 1 \leqslant k \leqslant K,$$

and compute the expectation vector E_k^i, by:

$$E_k^i = N\pi_k^i, \ 1 \leqslant k \leqslant K.$$

3: M-step: Re-estimate the parameters using the current responsibilities:

$$\pi_k^{i+1} = \frac{E_k^i + p(k|\mathbf{x}^*)}{N+1},$$

$$\boldsymbol{\mu}_k^{i+1} = \frac{E_k^i \boldsymbol{\mu}_k^i + \mathbf{x}^* p(k|\mathbf{x}^*)}{E_k^i + p(k|\mathbf{x}^*)},$$

$$\boldsymbol{\Sigma}_k^{i+1} = \frac{E_k^i[\boldsymbol{\Sigma}_k^i + (\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_k^{i+1})(\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_k^{i+1})^{\mathrm{T}}]}{E_k^i + p(k|\mathbf{x}^*)} + \frac{p(k|\mathbf{x}^*)(\mathbf{x}^* - \boldsymbol{\mu}_k^{i+1})(\mathbf{x}^* - \boldsymbol{\mu}_k^{i+1})^{\mathrm{T}}}{E_k^i + p(k|\mathbf{x}^*)}.$$

4: Evaluate the log-likelihood:

$$L(\Theta_{i+1}) = \sum_{k=1}^{K} p(k|\mathbf{x}^*)[ln\pi_k^{i+1} + ln\mathcal{N}(\mathbf{x}^*|\boldsymbol{\mu}_k^i, \boldsymbol{\Sigma}_k^i)].$$

5: $i \leftarrow i+1$.

6: **end while**

Algorithm 1 presents the main learning steps in the expectation–maximization algorithm for IGMM. The approach has separate code for the newly arrived data and for the data already fed to learn the model. The assumption is that the set of posterior probabilities $\{p(k|\mathbf{x}_j)\}_{j=1}^{N}$ remains the same when the model parameters are updated for the newly arrived data $\{\mathbf{x}^*\}$. By defining the newly arrived data $\{\mathbf{x}^*\}$ as an increase and the parameters of the learned model as $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$, the model parameters are updated after initialization via an iterative method with two steps: an expectation step (E-step) and a maximization step (M-step). In the E-step (line 2), the posterior probabilities of the component memberships are computed for the newly available data. It then computes the expectation vector for the next step. In the M-step (line 3), using the component-membership posterior probabilities as weights, the parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ are estimated based on maximizing the log-likelihood function. The E- and M-steps are constantly iterated until the log-likelihood function converges to the termination tolerance Δ, which is a small positive scalar (line 4).

Fig. 1 is a demo of how an IGMM learns from a data stream. The data stream has six different groups, each of which contains 100 data points (from [15]). From demos 1 to 6, the model is updated continually as the data stream is incrementally

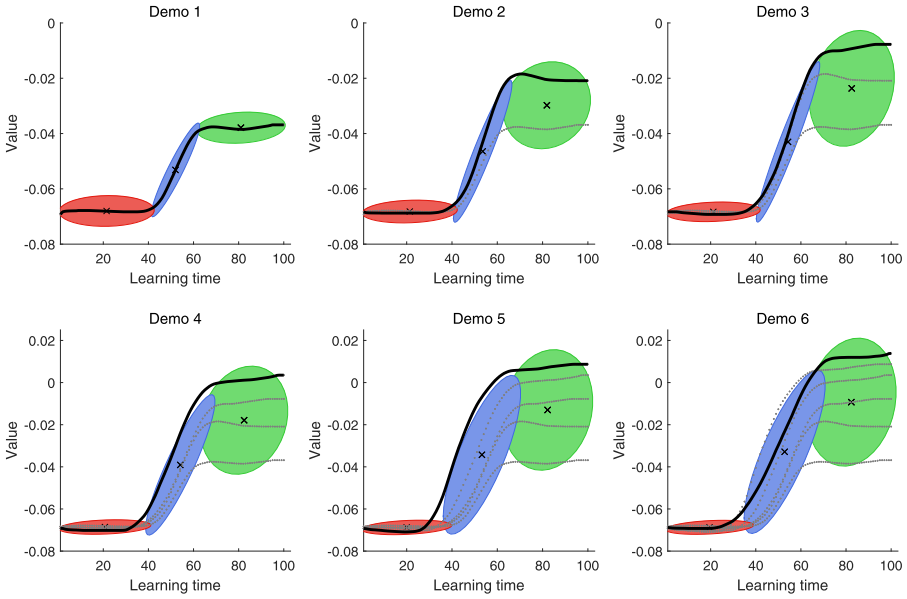


Fig. 1. The learning processes of IGMM.

fed into it. After the 6-th demo, the learning process has efficiently adapted the model to represent the data stream. Unlike an evolving data stream in an evolutionary search, the data stream used in the demo is independent and static. It has a global view of the data, i.e., all data points are randomly accessible. As the data stream is in a permuted order, the model does not change in the demo.

In contrast, the data in an evolving data stream depend on the generation and dynamically change. A window can be defined that slides over the evolving data stream, creating a partial view of the data. For a short window, the data are pseudo-stationary, whereas they converge over a longer time. Moreover, in IGMM, the data are stored and preserved during the learning process. An evolutionary search is an optimization process, and nondominated solutions gradually replace the worst solutions in the population during the evolutionary cycle. Hence, it is necessary to adjust IGMM if it is utilized to extract the structure of evolving data during an evolutionary process.

3. Proposed algorithm

The major motivation for ILME was to integrate an IGMM-based learning mechanism into MOEA, so that it obtains knowledge efficiently. The incremental-learning mechanism takes a stream of solutions sequentially generated during the evolution as training data to model the manifold structure of the PS and establish the neighborhood relationships among solutions. The parameters of the learning model are updated incrementally as newly generated data are collected to reduce the computational cost, unlike the batch approach that is currently used. With the learned model, the offspring production operation, which consists of a sampling strategy and a differential evolution operator, generates new trial solutions toward the whole PF. There is a good guarantee of population diversity in the objective space. An outline of our proposed ILME algorithm is shown in Fig. 2. The detailed steps are given as pseudocode in Algorithms 2–4.

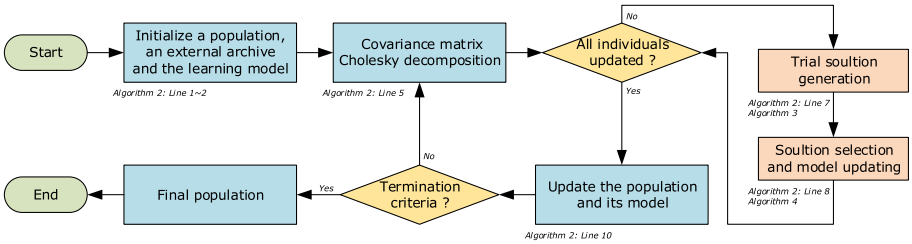


Fig. 2. Outline of the ILME.

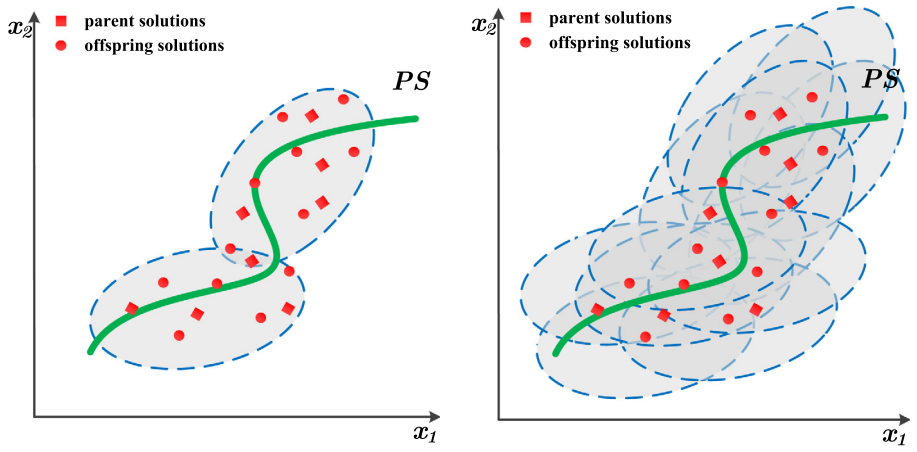
Algorithm 2. ILME Framework

Require: population size N, maximum evolutionary generations $\mathscr{T}$ and maximum number of components K_{max}.
Ensure: population $\mathscr{P}$.
1: Initialize a population $\mathscr{P} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^N\}$ randomly, and set the external archive $\mathscr{A} = \mathscr{P}$.
2: Let each $\mathbf{x}^i \in \mathscr{P}$as a component $\mathscr{C}_k$, update the $\mathscr{C}_k$ as follow:
$\boldsymbol{\mu}_k = \mathbf{x}^i$, $\boldsymbol{\Sigma}_k = \mathbf{I}$, $\pi_k = 1/N$.
3: **while** the termination criteria not met **do**
4: Count the number of components $K \leftarrow |\mathscr{C}_k|$.
5: Execute a Cholesky decomposition for covariance matrix of the each component $\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^{\mathsf{T}}, k = 1, 2, \ldots, K$.
6: **for** $i = 1$ to N **do**
7: Generate a trial solution $\mathbf{ts}^i \leftarrow Solgen(\boldsymbol{\Lambda}_k^i, \mathscr{C}_k, \mathbf{x}^i)$.
8: Selection and Updating $[\mathscr{A}, \Theta] \leftarrow Selupd(\mathscr{A}, \Theta, \mathbf{ts}^i)$.
9: **end for**
10: Update the population $\mathscr{P} = \mathscr{A}$ and pass the GMM parameter set Θ of $\mathscr{A}$ to $\mathscr{P}$.
11: **end while**

Pseudocode for ILME is given in Algorithm 2. In ILME, the population $\mathscr{P}$ is initialized randomly in the decision variable space. An external archive $\mathscr{A}$, built to maintain the solutions from the previous population, is assigned from $\mathscr{P}$ (line 1). Each solution is a component of the GMM in the initial population. The covariance matrix and weight coefficient are $\boldsymbol{\mu}_k = \mathbf{x}^i, \boldsymbol{\Sigma}_k = \mathbf{I}$, and $\pi_k = 1/N$, respectively (line 2). In the evolutionary cycle, first, the number of components is counted (line 4). For each component, a perturbation matrix $\boldsymbol{\Lambda}_k$ is generated using a Cholesky decomposition of its covariance matrix $\boldsymbol{\Sigma}_k$ (line 5). For each solution in the population, the proposed sampling strategy is utilized to create a trial solution (line 7). Then, the model parameter of the GMM is renewed incrementally with the solution as each new offspring solution is collected (line 8). When each solution in the current population has completed one evolution, the population $\mathscr{P}$ is renewed from the external archive $\mathscr{A}$ and a set of learned model parameters of the GMM is passed to it for the next evolution (line 10). The evolutionary cycles proceed until the termination criterion has been satisfied. Finally, ILME returns the final population.

3.1. Solution generation

The sampling strategy is a pivotal part of the trade-off between exploitation and exploration. A routine sampling strategy (RSS) in most EDAs is to add Gaussian noise after sampling from a learned model. Fig. 3(a) shows the search region for the RSS for a GMM with two components in an evaluation search. Fig. 3(a) shows 95% confidence regions for the parent solutions. The figure implies that the RSS can generate offspring solutions beyond the confidence region with a small probability (only 5%), which limits the search ability and readily leads to a loss of diversity in the population in the early stages of the evolution.



(a) Search regions of RSS (b) Search regions of SCSS

Fig. 3. Search regions under different sampling strategy. (a) Search region of RSS. (b) Search region of SCSS.

A simple, yet very effective way to overcome this inherent deficiency is to perform a Gaussian perturbation with the new trial solution in the current population. Knowledge from the learned model is used to guide the exploration. Hence, a novel sampling strategy is proposed, called the shared covariance-based advanced sampling strategy (SCSS). It uses information from the learned model. A new trial solution is perturbed using the covariance matrix of the component it belongs to. Fig. 3(b) shows that SCSS can explore beyond the current region, which certainly improves the search ability in exploration. Moreover, a differential evolution (DE) operator is used in the exploitation, especially in the last stage of the evolution. DE has two main control parameters which are the crossover constant (CR) and differentiation factor (F).

Algorithm 3. Solution Generation (SOLGEN) Operator

Require: a current solution $\mathbf{x}$, its perturbation matrix $\mathbf{\Lambda}$ and the component $\mathscr{C}_k$ it belongs to.
Ensure: a trial solution **ts**.
1: **if** $rand() < \varepsilon$
2: Sample $\mathbf{z} \in \mathbb{R}^n$ from $\mathcal{N}(0, \mathbf{I})$.
3: Generate a trial solution $\mathbf{ts} = \mathbf{x} + \mathbf{\Lambda z}$.
4: **else if**
5: Generate a trial solution **ts** using the DE operator, as follow:

$$ts_j = \begin{cases} x_j^i + F \times (x_j^{r_1} - x_j^{r_2}) & \text{if } r_1 < CR \\ x_j^i & \text{otherwise} \end{cases} \quad ,j = 1, 2, \ldots, n,$$

where $\mathbf{x}^{r_1}$ and $\mathbf{x}^{r_2}$ are parent solutions selected from $\mathscr{C}_k$, randomly.

6: **end if**
7: Repair the solution **ts**, as follow:

$$ts'_j = \begin{cases} a_j & \text{if } ts_j < a_j \\ b_j & \text{elseif } ts_j > b_j, \\ ts_j & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, n,$$

where a_j, b_j are the lower and upper bounds of the j-th variables, respectively.

8: Mutate the solution $\mathbf{ts}'$, as follow:

$$ts''_j = \begin{cases} ts'_j + \delta_j \times (b_j - a_j) & \text{if } r_2 < p_m \\ ts'_j & \text{otherwise} \end{cases} \quad ,j = 1, 2, \ldots, n,$$

with

$$\delta_j = \begin{cases} \left[2r_3 + (1 - 2r_3)\left(\frac{b_j - ts'_j}{b_j - a_j}\right)^{\eta_m + 1}\right]^{\frac{1}{\eta_m + 1}} - 1 & \text{if } r_3 < 0.5 \\ 1 - \left[2 - 2r_3 + (2r_3 - 1)\left(\frac{ts'_j - a_j}{b_j - a_j}\right)^{\eta_m + 1}\right]^{\frac{1}{\eta_m + 1}} & \text{otherwise} \end{cases}.$$

9: If necessary, repair the solution $\mathbf{ts} \leftarrow \mathbf{ts}''$.

Algorithm 3 shows the solution generation (SOLGEN) operator used in ILME. SCSS is utilized to produce a new trial solution with probability ε. Otherwise, the DE operator is used (lines 1–6). Then, a boundary repair operation is performed to ensure that the new offspring solution is feasible (line 7). Furthermore, the polynomial mutation operator is utilized on the feasible solution to improve the diversity of the population (line 8). Finally, the same boundary repair mechanism is performed again if necessary (line 9).

3.2. Selection and updating

In ILME, for each newly generated solution, first, its dominance relation is checked against each solution in the current external archive. The parameter of the learning model is updated for the new offspring solution only when it is not the worst. Hence, two operations are embedded within the selection and updating (SELUPD) operator in ILME. The superior selection mechanism is from SMS-EMOA [19] and the modification of the IGMM updating mechanism was inspired by agglomerative hierarchical clustering [40], in which pairs of clusters are merged successively when the number of clusters is above a threshold.

Algorithm 4. Selection & Updating (SELUPD) Operator

Require: a trial solution **ts**, the current external archive $\mathscr{A}$ and its model parameter set Θ.
Ensure: external archive $\mathscr{A}$ and its model parameter set Θ.
1: $\{\mathscr{R}^1, \mathscr{R}^2, \cdots, \mathscr{R}^L\} \leftarrow$ ***FastNondominatedSort*** $(\mathscr{A} \cup \{\mathbf{ts}\})$.
2: **if** $L > 1$
3: Determine $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}\in\mathscr{R}^L} d(\mathbf{x}, \mathscr{A} \cup \{\mathbf{ts}\})$.
4: **else if**
5: Determine $\mathbf{x}^* \leftarrow \arg\min_{\mathbf{x}\in\mathscr{R}^1} \Delta_\varphi(\mathbf{x}, \mathscr{R}^1)$.
6: **end if**
7: **if** $\mathbf{x}^* \neq \mathbf{ts}$
8: $\mathscr{A} \leftarrow \mathscr{A} \cup \{\mathbf{ts}\} \setminus \{\mathbf{x}^*\}$.
9: **if** $\mathscr{C}_k = \varnothing$
10: Delete the component $\mathscr{C}_k$, set $K \leftarrow K - 1$.
11: **else**
12: Update the component $\mathscr{C}_k$ as follow:

$$\pi_k = \frac{E_k - p(\mathbf{x}^*|k)}{N-1}, \quad \boldsymbol{\mu}_k = \frac{E_k\boldsymbol{\mu}_k - \mathbf{x}^* p(\mathbf{x}^*|k)}{E_k - p(\mathbf{x}^*|k)},$$

$$\boldsymbol{\Sigma}_k = \frac{E_k\boldsymbol{\Sigma}_k - p(\mathbf{x}^*|k)(\mathbf{x}^* - \boldsymbol{\mu}_k)(\mathbf{x}^* - \boldsymbol{\mu}_k)^{\mathrm{T}}}{E_k - p(\mathbf{x}^*|k)}.$$

13: **end if**
14: Set $K \leftarrow K + 1$, construct a new component $\mathscr{C}_K$, set $\pi_K = 1/N$, $\boldsymbol{\mu}_K = \mathbf{ts}$, $\boldsymbol{\Sigma}_k = \mathbf{I}$.
15: **if** $K > K_{max}$
16: Determine: $(\alpha, \beta) \leftarrow \arg\min_{\alpha,\beta,\alpha\neq\beta} \|\boldsymbol{\mu}_\alpha - \boldsymbol{\mu}_\beta\|$.
17: Merge the component as follows:
$\pi_\gamma = \pi_\alpha + \pi_\beta, \quad \boldsymbol{\mu}_\gamma = \frac{\pi_\alpha\boldsymbol{\mu}_\alpha + \pi_\beta\boldsymbol{\mu}_\beta}{\pi_\alpha + \pi_\beta}$,
$\boldsymbol{\Sigma}_\gamma = \frac{\pi_\alpha[\boldsymbol{\Sigma}_\alpha + (\boldsymbol{\mu}_\alpha - \boldsymbol{\mu}_\gamma)(\boldsymbol{\mu}_\alpha - \boldsymbol{\mu}_\gamma)^{\mathrm{T}}]}{\pi_\alpha + \pi_\beta} + \frac{\pi_\beta[\boldsymbol{\Sigma}_\beta + (\boldsymbol{\mu}_\beta - \boldsymbol{\mu}_\gamma)(\boldsymbol{\mu}_\beta - \boldsymbol{\mu}_\gamma)^{\mathrm{T}}]}{\pi_\alpha + \pi_\beta}$.
18: **end if**
19: **end if**

Details of the selection and updating (SELUPD) operator in ILME are shown in Algorithm 4. After the new trial solution **ts** is generated, the external archive $\mathscr{A}$ is updated as follows. First, the fast nondominated sorting method derived from NSGA-II [16] is utilized to partition the combined set $\mathscr{A} \cup \{\mathbf{ts}\}$ into L different fronts $\{\mathscr{R}^1, \mathscr{R}^2, \ldots, \mathscr{R}^L\}$, in which $\mathscr{R}^i$ is the i-th best front and $\mathscr{R}^L$ is the poorest one (line 1). Two strategies are used to determine a candidate solution $\mathbf{x}^*$, which is eliminated from the combined set: (1) Candidate solution $\mathbf{x}^*$ has the highest $d(\mathbf{x}, \mathscr{A} \cup \{\mathbf{ts}\})$ in $\mathscr{R}^L$ when there is more than one nondominated front in $\mathscr{A} \cup \{\mathbf{ts}\}$. Here, $d(\mathbf{x}, \mathscr{A} \cup \{\mathbf{ts}\})$ is the number of solutions that dominate $\mathbf{x}$ in $\mathscr{A} \cup \{\mathbf{ts}\}$ (line 3). (2) When there is only one nondominated front in $\mathscr{A} \cup \{\mathbf{ts}\}$, the candidate solution $\mathbf{x}^*$ has the lowest hypervolume (HV) $\Delta_\varphi(\mathbf{x}, \mathscr{R}^1)$ (line 5).

IF the trial solution **ts** is not a candidate solution $\mathbf{x}^*$, two operations are executed after the environmental selection: (1) eliminating the candidate solution and (2) modifying the parameter of the component. First, $\mathbf{x}^*$ is removed from the combined set (line 8). The parameter of the component it belongs to is updated, which is a decremental process (lines 9–13). Then, **ts** is used as a mean vector to construct a new component (line 14). If the GMM has more than K_{max} components, the two components with the shortest Euclidean distance between their mean vectors are merged into one, and the parameter is calculated for the merged component (lines 15–18).

3.3. Computational complexity

The computational cost of ILME is mainly due to the SELUPD operator. During the incremental learning, each datum is processed at most twice, once when it is processed as a new component that has survived and then if a merged component becomes too close to another component. In the worst case, a newly generated solution is nondominated and there are more than K_{max} components. The learning process requires $O(nK^2)$ computations to determine the two closest components. In the best case, the newly generated solution is utilized to construct a new component and no learning is required. In contrast,

batch learning, e.g., k-means clustering, requires $O(TKNn)$ computations. Here, T is the number of iterations. If $K \leqslant N$, then the computational cost of the proposed incremental-learning mechanism is less than $1/T$ times that for the batch k-means.

Next, we consider the time complexity of the SELUPD operator. First, the fast sorting requires $O(mN^2)$ computations. Second, there are two possible strategies for removing a solution, either by determining the number of solutions or by computing the HV. In the first strategy, the worst-case time complexity is $O(mN^2)$. The computational cost of the HV metric is exponential in m in the worst case, and the run time is $O(N^m)$ in our implementation. Third, merging the two closest components requires $O(nN^3)$ computations at the beginning of the evolutionary process, whereas it requires $O(nNK_{max}^2)$ computations to find the two closest clusters in almost all generations. In summary, the worst-case time complexity of ILME at each generation is $O(N^m + K^2Nn + mN^2)$, whereas the run time for the best case is $O(mN^2)$.

4. Numerical experiments

In this section, we describe the experimental design used to verify the performance of ILME on test instances (effectiveness and efficiency). We used six classical or newly developed MOEAs for comparison on three groups of 22 test instances with complicated characteristics. They are compared with performance metrics. Then, the parameter settings of the chosen algorithms are given. Finally, the experimental results are presented.

4.1. MOEAs for comparison and test instances

Six classical or newly developed MOEAs were chosen as competitive candidates to investigate empirically the performance of ILME. They span the major categories of MOEA in current efforts: (1) a Pareto dominance-based MOEA, NSGA-II [16], (2) a metric-based MOEA, SMS-EMOA [19], (3) a regularity model-based MOEDA, RM-MEDA [3], and (4) three state-of-the-art learning model-based MOEAs: SMEA [11], MOEA/D-LTD [2], and MDESL [39]. Of these algorithms, SMEA learns a SOM to establish a neighborhood to guide solution creation. MOEA/D-LTD has an analytical model based on a Gaussian process that adaptively sets the decomposition method by learning the characteristics of the estimated PF. MDESL applies k-means clustering to extract knowledge about the neighborhood relationship in the decision space for mating restriction.

The numerical experiments focus on continuous MOPs with complicated PF shapes. A total of 22 test instances were used to compare ILME with the other six MOEAs: the GLT test suites [41], the tri-objective test instances WFG1–WFG9 [42], and problems RE1–RE7 [43]. These test problems have various characteristics, such as the PF being convex, concave, mixed, disconnected, or degenerate, and the PS having multimodal, biased, deceptive, or nonlinear variable linkage. These pose a significant challenge to MOEA.

4.2. Performance metrics

Analyzing the performance of a MOEA essentially boils down to evaluating the approximate front obtained within some computational budget. The performance metrics are scalar quantities that reflect the quality of the scrutinized solution set with respect to some measure.

The averaged Hausdorff distance (Δ_p) [21] comprises the generational distance and the inverted generational distance. It is particularly useful for evaluating stochastic search algorithms. Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$ be finite and non-empty sets. Then, their averaged Hausdorff distance is defined as

$$\Delta_p(\mathbf{X}, \mathbf{Y}) = \max(\mathrm{GD}_p(\mathbf{X}, \mathbf{Y}), \mathrm{IGD}_p(\mathbf{X}, \mathbf{Y})) \tag{3}$$

where

$$\begin{aligned} \mathrm{GD}_p(\mathbf{X}, \mathbf{Y}) &= \left(\frac{1}{N} \sum_{i=1}^{N} \mathrm{dist}_p(\mathbf{x}_i, \mathbf{Y}) \right)^{1/p} \\ \mathrm{IGD}_p(\mathbf{X}, \mathbf{Y}) &= \left(\frac{1}{M} \sum_{i=1}^{M} \mathrm{dist}_p(\mathbf{y}_i, \mathbf{X}) \right)^{1/p} \end{aligned}$$

and $\mathrm{dist}_p(\mathbf{x}_i, \mathbf{Y})$ is the minimal distance from $\mathbf{x}_i \in \mathbf{X}$ to $\mathbf{Y}$:

$$\mathrm{dist}_p(\mathbf{x}_i, \mathbf{Y}) = \inf_{y \in \mathbf{Y}} \|\mathbf{x}_i - y\|_p.$$

In this paper, we choose $p = 2$. Let $\mathscr{P}^*$ be a set of uniformly sampled points in the true PF, and $\mathscr{P}$ be a set of nondominated solutions obtained by an MOEA. We compute $\Delta_2(\mathscr{P}^*, \mathscr{P})$ to assess its performance. The smaller Δ_2 is, the better $\mathscr{P}$ is.

The Hypervolume (HV) metric [44,45] measures the volume covered by nondominated solutions. It is generally the region bounded by all points in the approximate front $\mathscr{P}$ with respect to the reference point $\mathbf{r}^*$. Mathematically, it is defined as:

$$\mathrm{HV}(\mathscr{P}, \mathbf{r}^*) = \mathrm{VOL}\left(\bigcup_{\mathbf{x}\in\mathscr{P}} [f_1(\mathbf{x}), \mathbf{r}_1] \times \ldots \times [f_m(\mathbf{x}), \mathbf{r}_m]\right) \tag{4}$$

where $\mathbf{r}^* = (r_1, \ldots, r_m)$ is a reference point, which must be dominated by any point in the PF, and $\mathrm{VOL}(\cdot)$ is the Lebesgue measure. A larger HV value is desirable, as it reflects that the generated approximate front is close to the PF.

4.3. Parameter settings

Different algorithmic descriptions and software implementations can lead to statistically significant differences in the performance of the MOEAs [46]. In order to ensure that different algorithms can be compared in a fair environment, all algorithms are implemented in PlatEMO [47], and the parameters for the algorithms compared were set to the recommended values as reported in their original publications. The DE operator and polynomial mutation [48] were adopted for offspring generation in NSGA-II and SMS-EMOA as suggested in [38]. The reference points were predefined as: $\mathbf{r} = (2, 2)^{\mathrm{T}}$ for GLT1 and GLT3, $\mathbf{r} = (2, 11)^{\mathrm{T}}$ for GLT2, $\mathbf{r} = (2, 3)^{\mathrm{T}}$ for GLT4, $\mathbf{r} = (2, 2, 2)^{\mathrm{T}}$ for GLT5 and GLT6, $\mathbf{r} = (3, 5, 7)^{\mathrm{T}}$ for WFG1–WFG9. Note that the recommended settings of the RE problems are used in the numerical experiments. The detailed parameter settings for the different algorithms are listed in Table 1.

4.4. Experimental results

In the numerical experiments, each algorithm was run independently 31 times on each test instance. To quantify intuitively the performance of each algorithm, the mean and standard deviation (*sd*) were calculated for each metric. These are listed in the relevant results table (Tables 2–4). The algorithms were ranked for each test instance in ascending (descending) order of Δ_p (HV). The ranks are shown in square brackets in the tables. The standard deviations are given as subscripts. The Wilcoxon rank–sum (Mann–Whitney) test based on a 5% significance level was conducted with the mean values of each metric to illustrate the statistical differences. The labels $\dagger, \S$, and $\approx$ in the tables denote whether ILME is superior, inferior, or similar to that of the algorithm it is compared to, respectively, in terms of the Wilcoxon rank–sum test. Further, the mean ranking, which is a comprehensive indicator for evaluating the performance of an algorithm, was also calculated. The performance of ILME is compared with the other algorithms on the GLT test suites in terms of convergence quality and convergence speed (Fig. 4). The results are visualized in Figs. 5 and 6.

Table 1
Parameter setting of all the compared algorithms.

MOEAs	Parameter Values
Public Parameters	Population size: $N = 100(105)$ for bi-objective (tri-objective) MOPs; Variable dimension: $n = 10$ for GLT, $n = 12$ for WFG; Maximum evolutionary generations: $\mathscr{T} = 300$; Executed times: 31 independent runs on each test instance; Stopping criterion: all algorithms terminate execution after 100,000 function evaluations.
ILME	Maximum number of components: $K_{max} = 7$; Exploitation probability: $\varepsilon = 0.7$; Crossover and differentiation constant: $CR = 1$, $F = 0.7$; Mutation operator control parameters: $p_m = 1/n$, $\eta_m = 20$.
SMEA	Structures of SOM: 1 (2)-D structure $1 \times 100(7 \times 15)$ for bi-objective (tri-objective) MOPs; Initial learning rate: $\tau_0 = 0.9$; Neighborhood size: $T = 5$; Probability of mating restriction: $\beta = 0.7$; Maximal number of replacement: $n_r = 1$.
MDESL	Maximum number of clusters: $K = 10$; History length: $H = 5$; Crossover and differentiation constant: $F_t = 0.6, CR_t = 1; F_r = 0.6, CR_r = 1$.
MOEA/D-LTD	Neighborhood size: $T = 20$; Probability of neighborhood search: $\beta = 0.9$; Begin(end) percentage, interval of performing LTD procedure: $\psi_b = 50\%(\psi_e = 50\%)$, $\tau = 20$; Rate of sampling to the population: $\phi = 40$; Threshold of variance coefficient: $V_t = 1.4$; Maximal number of replaced solutions: $n_r = 2$.
RM-MEDA	Maximal number of iteration in local PCA: $T = 50$; Number of clusters in local PCA: $K = 5$; Extension rate of sampling: $\phi = 0.25$.
NSGA-II and SMS-EMOA	Crossover and differentiation constant: $CR = 1$, $F = 0.5$; Mutation operator control parameters: $p_m = 1/n$, $\eta_m = 20$.

Table 2
Mean and *sd* of Δ_2 metric after 31 independent runs of seven algorithms on the 22 test instances.

Instance	NSGA-II	SMS-EMOA	RM-MEDA	SMEA	MDSEL	MOEA/D-LTD	ILME
				Δ_2			
GLT1	$1.643e{-}02^{\dagger}_{1.02e-02}$[6]	$1.901e{-}02^{\dagger}_{1.03e-02}$[7]	$1.519e{-}02^{\dagger}_{1.30e-02}$[5]	$6.435e{-}03^{\dagger}_{1.01e-03}$[4]	$5.314e{-}03^{\dagger}_{7.95e-04}$[2]	$5.636e{-}03^{\dagger}_{4.41e-04}$[3]	$1.915e{-}03_{5.47e-05}$[1]
GLT2	$4.236e{-}02^{\dagger}_{3.13e-03}$[6]	$4.577e{-}02^{\dagger}_{5.42e-03}$[7]	$3.363e{-}02^{\S}_{1.19e-03}$[1]	$3.503e{-}02^{\S}_{3.11e-03}$[2]	$3.569e{-}02^{\dagger}_{5.43e-03}$[4]	$3.615e{-}02^{\dagger}_{1.19e-03}$[5]	$3.556e{-}02_{1.13e-03}$[3]
GLT3	$1.364e{-}02^{\dagger}_{8.18e-03}$[4]	$2.810e{-}02^{\dagger}_{1.40e-02}$[6]	$2.045e{-}02^{\dagger}_{1.27e-02}$[5]	$7.020e{-}03^{\dagger}_{5.06e-04}$[2]	$4.529e{-}02^{\dagger}_{8.65e-02}$[7]	$8.569e{-}03^{\dagger}_{2.26e-03}$[3]	$4.993e{-}03_{8.26e-05}$[1]
GLT4	$4.851e{-}02^{\dagger}_{5.78e-02}$[6]	$5.697e{-}02^{\dagger}_{4.99e-02}$[7]	$4.191e{-}02^{\dagger}_{4.79e-02}$[5]	$1.227e{-}02^{\dagger}_{1.33e-03}$[3]	$2.740e{-}02^{\dagger}_{5.91e-02}$[4]	$1.168e{-}02^{\dagger}_{1.44e-03}$[2]	$5.665e{-}03_{6.87e-05}$[1]
GLT5	$6.304e{-}02^{\dagger}_{4.29e-03}$[7]	$3.014e{-}02^{\dagger}_{4.62e-04}$[2]	$5.163e{-}02^{\dagger}_{1.93e-03}$[6]	$4.492e{-}02^{\dagger}_{1.20e-03}$[4]	$4.420e{-}02^{\dagger}_{9.00e-04}$[3]	$4.497e{-}02^{\dagger}_{1.25e-03}$[5]	$2.902e{-}02_{3.26e-04}$[1]
GLT6	$5.460e{-}02^{\dagger}_{4.50e-03}$[7]	$2.283e{-}02^{\approx}_{1.30e-03}$[1]	$3.816e{-}02^{\dagger}_{1.70e-03}$[6]	$3.374e{-}02^{\dagger}_{2.93e-03}$[5]	$3.248e{-}02^{\dagger}_{7.94e-04}$[3]	$3.350e{-}02^{\dagger}_{9.51e-04}$[4]	$2.416e{-}02_{2.61e-03}$[2]
Mean Rank	6.000	5.000	4.667	3.333	3.833	3.667	1.500
†/§/≈	6/0/0	5/0/1	5/1/0	5/1/0	6/0/0	6/0/0	
WFG1	$1.238e{+}00^{\S}_{3.85e-03}$[4]	$1.213e{+}00^{\S}_{5.62e-03}$[2]	$1.227e{+}00^{\S}_{9.02e-03}$[3]	$1.523e{+}00^{\dagger}_{6.61e-02}$[7]	$1.040e{+}00^{\S}_{1.22e-02}$[1]	$1.428e{+}00^{\approx}_{5.87e-02}$[5]	$1.444e{+}00_{4.50e-02}$[6]
WFG2	$5.337e{-}02^{\dagger}_{3.74e-03}$[7]	$4.174e{-}02^{\dagger}_{1.59e-03}$[6]	$4.042e{-}02^{\dagger}_{7.57e-04}$[5]	$1.114e{-}02^{\approx}_{5.25e-04}$[1]	$3.824e{-}02^{\dagger}_{4.51e-04}$[4]	$1.748e{-}02^{\dagger}_{1.54e-02}$[3]	$1.123e{-}02_{4.42e-04}$[2]
WFG3	$3.164e{-}02^{\dagger}_{2.08e-03}$[7]	$2.603e{-}02^{\dagger}_{1.72e-03}$[6]	$2.021e{-}02^{\dagger}_{1.09e-03}$[5]	$1.135e{-}02^{\approx}_{1.02e-04}$[2]	$1.716e{-}02^{\dagger}_{7.39e-04}$[4]	$1.202e{-}02^{\dagger}_{2.12e-03}$[3]	$1.131e{-}02_{9.15e-05}$[1]
WFG4	$1.062e{-}01^{\dagger}_{5.84e-03}$[7]	$1.031e{-}01^{\dagger}_{1.06e-02}$[6]	$7.882e{-}02^{\dagger}_{1.12e-02}$[5]	$1.009e{-}02^{\approx}_{2.90e-04}$[2]	$2.423e{-}02^{\dagger}_{2.56e-03}$[4]	$1.095e{-}02^{\dagger}_{1.41e-03}$[3]	$1.003e{-}02_{1.94e-04}$[1]
WFG5	$7.084e{-}02^{\dagger}_{2.27e-03}$[7]	$6.847e{-}02^{\dagger}_{5.70e-04}$[4]	$6.879e{-}02^{\dagger}_{9.69e-04}$[6]	$6.634e{-}02^{\approx}_{9.26e-04}$[2]	$6.732e{-}02^{\dagger}_{3.50e-04}$[3]	$6.855e{-}02^{\dagger}_{4.34e-03}$[5]	$6.610e{-}02_{1.18e-03}$[1]
WFG6	$3.585e{-}01^{\dagger}_{2.70e-02}$[3]	$3.780e{-}01^{\dagger}_{1.03e-02}$[7]	$3.676e{-}01^{\dagger}_{2.31e-02}$[5]	$3.519e{-}01^{\approx}_{2.69e-02}$[2]	$3.654e{-}01^{\dagger}_{1.98e-02}$[4]	$3.774e{-}01^{\dagger}_{2.02e-02}$[6]	$3.394e{-}01_{2.33e-02}$[1]
WFG7	$2.777e{-}02^{\dagger}_{1.32e-03}$[7]	$2.402e{-}02^{\dagger}_{1.63e-03}$[6]	$2.060e{-}02^{\dagger}_{7.87e-04}$[5]	$1.065e{-}02^{\approx}_{2.50e-04}$[2]	$1.836e{-}02^{\dagger}_{4.42e-04}$[4]	$1.753e{-}02^{\dagger}_{8.96e-03}$[3]	$1.062e{-}02_{2.90e-04}$[1]
WFG8	$7.166e{-}02^{\dagger}_{7.73e-03}$[6]	$7.976e{-}02^{\dagger}_{1.67e-02}$[7]	$5.746e{-}02^{\dagger}_{1.59e-02}$[5]	$1.325e{-}02^{\dagger}_{7.60e-04}$[2]	$4.875e{-}02^{\dagger}_{8.79e-03}$[4]	$2.074e{-}02^{\dagger}_{1.11e-02}$[3]	$1.238e{-}02_{5.16e-04}$[1]
WFG9	$2.537e{-}01^{\dagger}_{2.65e-02}$[6]	$2.201e{-}01^{\dagger}_{2.08e-03}$[2]	$2.341e{-}01^{\dagger}_{2.77e-02}$[5]	$2.238e{-}01^{\approx}_{6.44e-02}$[4]	$2.722e{-}01^{\dagger}_{3.86e-05}$[7]	$2.216e{-}01^{\dagger}_{4.06e-02}$[3]	$1.995e{-}01_{4.60e-03}$[1]
Mean Rank	6.000	5.111	4.889	2.667	3.889	3.778	1.667
†/§/≈	8/1/0	8/1/0	8/1/0	2/0/7	8/1/0	8/0/1	
RE2-4-1	$1.882e{-}01^{\dagger}_{4.44e-02}$[6]	$2.634e{-}02^{\dagger}_{1.10e-02}$[4]	$1.324e{-}02^{\dagger}_{2.81e-03}$[3]	$1.032e{-}02^{\S}_{1.39e-04}$[1]	$9.697e{-}02^{\dagger}_{8.89e-04}$[5]	$3.273e{-}01^{\S}_{2.07e-02}$[7]	$1.090e{-}02_{3.58e-04}$[2]
RE2-3-2	$4.899e{-}01^{\approx}_{6.84e-01}$[5]	$4.467e{-}01^{\approx}_{3.87e-01}$[3]	$5.204e{-}01^{\approx}_{2.59e-01}$[6]	$1.336e{-}01^{\S}_{5.57e-02}$[2]	$4.772e{-}02^{\S}_{8.51e-02}$[1]	$2.484e{+}00^{\S}_{2.41e+00}$[7]	$4.500e{-}01_{1.95e-01}$[4]
RE2-4-3	$4.424e{-}01^{\dagger}_{4.91e-01}$[6]	$2.578e{-}01^{\approx}_{2.50e-01}$[3]	$2.965e{-}01^{\approx}_{4.96e-01}$[5]	$2.379e{-}02^{\S}_{1.50e-02}$[1]	$6.149e{-}02^{\approx}_{2.37e-03}$[2]	$9.491e{-}01^{\S}_{9.80e-02}$[7]	$2.808e{-}01_{4.05e-01}$[4]
RE2-2-4	$1.014e{-}02^{\S}_{1.03e-04}$[1]	$1.404e{-}02^{\dagger}_{2.19e-04}$[4]	$1.994e{-}02^{\dagger}_{1.42e-03}$[5]	$8.920e{-}01^{\dagger}_{2.04e-01}$[6]	$1.029e{-}02^{\S}_{2.71e-05}$[2]	$1.350e{+}00^{\dagger}_{1.74e-01}$[7]	$1.259e{-}02_{5.70e-04}$[3]
RE2-3-5	$1.075e{-}01^{\dagger}_{7.97e-04}$[5]	$4.841e{-}02^{\dagger}_{1.97e-03}$[3]	$4.496e{-}02^{\dagger}_{2.06e-03}$[2]	$2.933e{-}01^{\dagger}_{2.40e-02}$[6]	$8.920e{-}02^{\dagger}_{6.25e-04}$[4]	$4.782e{-}01^{\dagger}_{3.07e-02}$[7]	$3.599e{-}02_{1.19e-03}$[1]
RE3-3-1	$7.273e{-}01^{\approx}_{1.79e-01}$[4]	$8.073e{-}01^{\dagger}_{1.37e-01}$[5]	$4.814e{-}01^{\S}_{1.55e-01}$[2]	$8.435e{-}01^{\dagger}_{9.57e-02}$[6]	$9.080e{-}02^{\S}_{1.92e-02}$[1]	$1.100e{+}00^{\dagger}_{1.41e-02}$[7]	$6.587e{-}01_{1.18e-01}$[3]
RE3-4-2	$3.550e{-}02^{\S}_{4.91e-04}$[2]	$5.054e{-}02^{\approx}_{1.63e-03}$[3]	$9.604e{-}02^{\dagger}_{1.58e-02}$[5]	$1.230e{+}01^{\dagger}_{2.06e+00}$[6]	$3.089e{-}02^{\S}_{3.57e-04}$[1]	$2.119e{+}01^{\dagger}_{3.19e+00}$[7]	$5.068e{-}02_{4.31e-03}$[4]
Mean Rank	4.143	3.571	4.000	4.000	2.286	7.000	3.000
†/§/≈	3/2/2	4/0/3	4/1/2	4/3/0	2/4/1	7/0/0	
Overall Mean Rank	5.409	4.591	4.545	3.273	3.364	4.773	2.045
†/§/≈	17/3/2	17/1/4	17/3/2	11/4/7	16/5/1	21/0/1	

Table 3
Mean and *sd* of HV metric after 31 independent runs of seven algorithms on the 22 test instances.

Instance	NSGA-II	SMS-EMOA	RM-MEDA	SMEA	MDSEL	MOEA/D-LTD	ILME
				HV			
GLT1	3.309e+00$^{\dagger}_{2.41e-02}$[6]	3.300e+00$^{\dagger}_{2.58e-02}$[7]	3.311e+00$^{\dagger}_{3.38e-02}$[5]	3.365e+00$^{\S}_{2.01e-03}$[4]	3.366e+00$^{\S}_{2.06e-03}$[3]	3.366e+00$^{\S}_{2.27e-03}$[2]	3.371e+00$_{1.35e-03}$[1]
GLT2	1.972e+01$^{\dagger}_{9.98e-03}$[6]	1.975e+01$^{\dagger}_{1.85e-02}$[5]	1.970e+01$^{\dagger}_{6.99e-03}$[7]	1.978e+01$^{\S}_{1.21e-02}$[2]	1.978e+01$^{\S}_{2.49e-02}$[3]	1.976e+01$^{\S}_{4.87e-03}$[4]	1.981e+01$_{7.32e-04}$[1]
GLT3	3.946e+00$^{\dagger}_{1.56e-03}$[4]	3.943e+00$^{\dagger}_{2.57e-03}$[6]	3.944e+00$^{\dagger}_{2.10e-03}$[5]	3.948e+00$^{\S}_{3.33e-04}$[2]	3.935e+00$^{\S}_{2.87e-02}$[7]	3.947e+00$^{\S}_{4.19e-04}$[3]	3.949e+00$_{9.22e-05}$[1]
GLT4	4.943e+00$^{\dagger}_{9.52e-02}$[5]	4.936e+00$^{\dagger}_{5.14e-02}$[7]	4.962e+00$^{\dagger}_{3.11e-02}$[4]	4.982e+00$^{\S}_{3.84e-03}$[3]	4.938e+00$^{\S}_{1.79e-01}$[6]	4.983e+00$^{\S}_{3.03e-03}$[2]	4.993e+00$_{3.70e-04}$[1]
GLT5	7.939e+00$^{\dagger}_{3.15e-03}$[7]	7.968e+00$^{\dagger}_{1.99e-04}$[2]	7.952e+00$^{\dagger}_{1.33e-03}$[6]	7.957e+00$^{\S}_{1.48e-03}$[4]	7.958e+00$^{\S}_{7.12e-04}$[3]	7.956e+00$^{\S}_{8.56e-04}$[5]	7.969e+00$_{1.01e-04}$[1]
GLT6	7.933e+00$^{\dagger}_{3.45e-03}$[7]	7.960e+00$^{\dagger}_{1.18e-03}$[2]	7.948e+00$^{\dagger}_{1.24e-03}$[6]	7.949e+00$^{\S}_{2.89e-03}$[5]	7.952e+00$^{\S}_{1.14e-03}$[3]	7.951e+00$^{\S}_{7.68e-04}$[4]	7.961e+00$_{2.46e-03}$[1]
Mean Rank	5.833	4.833	5.500	3.333	4.167	3.333	1.000
†/§/≈	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	
WFG1	5.325e+00$^{\approx}_{1.76e-02}$[4]	5.453e+00$^{\S}_{2.46e-02}$[2]	5.378e+00$^{\approx}_{3.56e-02}$[3]	4.742e+00$^{\S}_{4.60e-01}$[7]	6.119e+00$^{\dagger}_{6.48e-02}$[1]	5.323e+00$^{\approx}_{4.00e-01}$[5]	5.293e+00$_{2.93e-01}$[6]
WFG2	1.125e+01$^{\dagger}_{2.94e-02}$[7]	1.138e+01$^{\dagger}_{7.56e-03}$[5]	1.139e+01$^{\dagger}_{6.91e-03}$[4]	1.146e+01$^{\S}_{8.31e-04}$[2]	1.144e+01$^{\S}_{1.26e-03}$[3]	1.135e+01$^{\S}_{2.50e-01}$[6]	1.146e+01$_{3.38e-05}$[1]
WFG3	1.078e+01$^{\dagger}_{1.48e-02}$[7]	1.081e+01$^{\dagger}_{1.45e-02}$[6]	1.086e+01$^{\dagger}_{8.51e-03}$[5]	1.096e+01$^{\S}_{8.27e-05}$[2]	1.090e+01$^{\S}_{6.36e-03}$[4]	1.095e+01$^{\S}_{2.24e-02}$[3]	1.096e+01$_{9.72e-05}$[1]
WFG4	8.051e+00$^{\dagger}_{3.98e-02}$[7]	8.148e+00$^{\dagger}_{5.32e-02}$[6]	8.254e+00$^{\dagger}_{6.14e-02}$[5]	8.683e+00$^{\approx}_{3.61e-03}$[2]	8.588e+00$^{\S}_{1.77e-02}$[4]	8.668e+00$^{\S}_{1.52e-02}$[3]	8.685e+00$_{3.15e-03}$[1]
WFG5	8.196e+00$^{\approx}_{5.24e-02}$[3]	8.156e+00$^{\dagger}_{5.12e-02}$[6]	8.228e+00$^{\S}_{5.28e-02}$[1]	8.161e+00$^{\approx}_{4.36e-02}$[5]	8.211e+00$^{\dagger}_{5.31e-02}$[2]	8.138e+00$^{\S}_{2.51e-02}$[7]	8.172e+00$_{5.25e-02}$[4]
WFG6	6.218e+00$^{\dagger}_{1.35e-01}$[3]	6.121e+00$^{\dagger}_{5.21e-02}$[7]	6.173e+00$^{\dagger}_{1.16e-01}$[5]	6.253e+00$^{\approx}_{1.36e-01}$[2]	6.183e+00$^{\S}_{9.97e-02}$[4]	6.125e+00$^{\S}_{1.00e-01}$[6]	6.317e+00$_{1.18e-01}$[1]
WFG7	8.546e+00$^{\dagger}_{1.09e-02}$[7]	8.587e+00$^{\dagger}_{1.21e-02}$[6]	8.606e+00$^{\dagger}_{8.16e-03}$[5]	8.688e+00$^{\dagger}_{8.70e-04}$[1]	8.648e+00$^{\S}_{3.63e-03}$[3]	8.618e+00$^{\S}_{6.78e-02}$[4]	8.688e+00$_{1.88e-03}$[2]
WFG8	8.258e+00$^{\dagger}_{4.87e-02}$[6]	8.244e+00$^{\dagger}_{9.86e-02}$[7]	8.363e+00$^{\dagger}_{8.98e-02}$[5]	8.667e+00$^{\S}_{7.47e-03}$[2]	8.476e+00$^{\S}_{4.57e-02}$[4]	8.590e+00$^{\S}_{8.33e-02}$[3]	8.672e+00$_{9.89e-03}$[1]
WFG9	6.148e+00$^{\dagger}_{1.20e-01}$[6]	6.276e+00$^{\dagger}_{2.55e-02}$[4]	6.265e+00$^{\dagger}_{1.38e-01}$[5]	6.339e+00$^{\approx}_{3.06e-01}$[3]	6.100e+00$^{\S}_{1.62e-02}$[7]	6.356e+00$^{\S}_{1.89e-01}$[2]	6.453e+00$_{3.49e-02}$[1]
Mean Rank	5.556	5.444	4.222	2.889	3.556	4.333	2.000
†/§/≈	7/0/2	8/1/0	7/1/1	4/1/4	7/2/0	8/0/1	
RE2-4-1	8.547e−01$^{\dagger}_{4.32e-04}$[5]	8.534e−01$^{\dagger}_{1.97e-03}$[6]	8.559e−01$^{\dagger}_{6.45e-05}$[3]	8.560e−01$^{\S}_{2.46e-05}$[1]	8.556e−01$^{\dagger}_{1.39e-05}$[4]	7.825e−01$^{\dagger}_{3.80e-03}$[7]	8.560e−01$_{2.91e-05}$[2]
RE2-3-2	6.548e−01$^{\dagger}_{1.62e-01}$[6]	6.767e−01$^{\S}_{1.05e-01}$[3]	6.690e−01$^{\approx}_{4.24e-02}$[5]	7.470e−01$^{\S}_{6.97e-03}$[2]	7.591e−01$^{\S}_{8.07e-03}$[1]	3.429e−01$^{\dagger}_{1.54e-01}$[7]	6.760e−01$_{2.98e-02}$[4]
RE2-4-3	1.148e+00$^{\dagger}_{1.48e-02}$[5]	1.137e+00$^{\dagger}_{1.93e-02}$[6]	1.152e+00$^{\dagger}_{1.50e-02}$[4]	1.160e+00$^{\S}_{1.13e-03}$[2]	1.160e+00$^{\approx}_{9.61e-05}$[1]	9.919e−01$^{\dagger}_{1.28e-02}$[7]	1.153e+00$_{1.22e-02}$[3]
RE2-2-4	1.169e+00$^{\S}_{1.06e-03}$[2]	1.147e+00$^{\dagger}_{1.39e-03}$[5]	1.151e+00$^{\approx}_{2.72e-03}$[4]	1.140e+00$^{\dagger}_{3.06e-02}$[6]	1.175e+00$^{\S}_{1.15e-04}$[1]	0.000e+00$^{\S}_{0.00e+00}$[7]	1.153e+00$_{2.70e-03}$[3]
RE2-3-5	1.077e+00$^{\dagger}_{7.09e-05}$[5]	1.079e+00$^{\dagger}_{1.03e-04}$[2]	1.079e+00$^{\dagger}_{1.33e-04}$[3]	1.040e+00$^{\dagger}_{3.28e-03}$[6]	1.078e+00$^{\dagger}_{7.74e-05}$[4]	1.006e+00$^{\S}_{5.14e-03}$[7]	1.080e+00$_{4.96e-05}$[1]
RE3-3-1	9.490e−01$^{\approx}_{8.87e-02}$[4]	8.982e−01$^{\dagger}_{6.48e-02}$[5]	1.067e+00$^{\S}_{8.40e-02}$[2]	8.916e−01$^{\dagger}_{5.41e-02}$[6]	1.330e+00$^{\S}_{1.54e-02}$[1]	7.745e−01$^{\S}_{1.31e-02}$[7]	9.678e−01$_{5.77e-02}$[3]
RE3-4-2	1.332e+00$^{\S}_{1.23e-03}$[1]	1.319e+00$^{\approx}_{1.49e-03}$[3]	1.301e+00$^{\dagger}_{4.05e-03}$[5]	1.291e+00$^{\dagger}_{3.13e-02}$[6]	1.327e+00$^{\S}_{3.52e-04}$[2]	1.290e+00$^{\dagger}_{2.25e-02}$[7]	1.318e+00$_{2.98e-03}$[4]
Mean Rank	4.000	4.286	3.714	4.143	2.000	7.000	2.857
†/§/≈	4/2/1	5/1/1	4/1/2	4/3/0	2/4/1	7/0/0	
Overall Mean Rank	5.136	4.909	4.409	3.409	3.227	4.909	2.182
†/§/≈	17/2/3	19/2/1	17/2/3	14/4/4	15/6/1	21/0/1	

Table 4
Mean and *sd* of run times of seven algorithms over 31 independent runs on the 22 test instances.

Instance	NSGA-II	SMS-EMOA	RM-MEDA	SMEA	MDSEL	MOEA/D-LTD	ILME
GLT1	2.665e+00$_{6.40e-01}$[1]	1.376e+01$_{1.22e+00}$[3]	6.004e+00$_{6.52e-01}$[2]	2.254e+01$_{2.66e+00}$[7]	1.674e+01$_{2.96e+00}$[5]	1.851e+01$_{1.11e+00}$[6]	1.400e+01$_{2.35e+00}$[4]
GLT2	1.670e+00$_{3.21e-01}$[1]	1.167e+01$_{2.65e-01}$[3]	5.447e+00$_{1.14e+00}$[2]	1.916e+01$_{1.81e+00}$[6]	1.476e+01$_{2.19e+00}$[5]	1.926e+01$_{2.22e-01}$[7]	1.362e+01$_{2.44e+00}$[4]
GLT3	1.348e+00$_{4.60e-02}$[1]	1.194e+01$_{3.57e-01}$[3]	3.767e+00$_{1.48e-01}$[2]	1.905e+01$_{1.13e+00}$[6]	1.389e+01$_{1.75e+00}$[5]	1.958e+01$_{1.16e+00}$[7]	1.321e+01$_{1.27e+00}$[4]
GLT4	2.460e+00$_{9.17e-01}$[1]	1.172e+01$_{3.17e-01}$[3]	4.666e+00$_{5.47e-01}$[2]	1.921e+01$_{4.28e-01}$[7]	1.374e+01$_{1.82e+00}$[5]	1.901e+01$_{9.38e-01}$[6]	1.173e+01$_{1.26e+00}$[4]
GLT5	1.567e+00$_{2.54e-01}$[1]	3.560e+02$_{4.58e+01}$[7]	4.961e+00$_{5.06e-01}$[2]	1.399e+02$_{4.20e+00}$[6]	1.369e+01$_{1.80e+00}$[4]	2.905e+01$_{1.32e+00}$[5]	1.096e+01$_{1.20e+00}$[3]
GLT6	1.589e+00$_{7.73e-02}$[1]	3.808e+02$_{4.40e+01}$[7]	4.538e+00$_{3.44e-01}$[2]	1.967e+02$_{3.34e+01}$[6]	1.479e+01$_{1.76e+00}$[4]	2.940e+01$_{1.25e+00}$[5]	1.197e+01$_{1.40e+00}$[3]
WFG1	1.721e+00$_{5.67e-01}$[1]	2.481e+02$_{1.76e+01}$[7]	5.073e+00$_{7.46e-01}$[2]	8.187e+01$_{8.24e+00}$[6]	2.074e+01$_{4.16e+00}$[4]	3.412e+01$_{2.00e+00}$[5]	1.487e+01$_{2.07e+00}$[3]
WFG2	1.747e+00$_{4.76e-01}$[1]	3.698e+02$_{2.14e+01}$[7]	5.569e+00$_{6.91e-01}$[2]	8.860e+01$_{5.68e+00}$[6]	2.216e+01$_{5.15e+00}$[4]	3.588e+01$_{1.41e+00}$[5]	1.533e+01$_{2.20e+00}$[3]
WFG3	1.399e+00$_{8.10e-02}$[1]	5.133e+02$_{6.04e+01}$[7]	6.955e+00$_{5.43e-01}$[2]	2.775e+02$_{3.29e+01}$[6]	1.694e+01$_{2.35e+00}$[4]	4.925e+01$_{2.01e+00}$[5]	1.419e+01$_{1.88e+00}$[3]
WFG4	1.328e+00$_{1.13e-01}$[1]	4.073e+02$_{5.89e+01}$[7]	7.594e+00$_{9.40e-01}$[2]	1.408e+02$_{1.73e+01}$[6]	1.533e+01$_{2.01e+00}$[4]	4.330e+01$_{1.32e+00}$[5]	1.341e+01$_{1.80e+00}$[3]
WFG5	1.268e+00$_{1.21e-01}$[1]	3.946e+02$_{4.09e+01}$[7]	5.478e+00$_{5.09e-01}$[2]	2.202e+02$_{2.53e+01}$[6]	1.535e+01$_{1.61e+00}$[4]	4.457e+01$_{1.15e+00}$[5]	1.399e+01$_{1.89e+00}$[3]
WFG6	1.327e+00$_{1.35e-01}$[1]	3.145e+02$_{4.26e+01}$[7]	5.749e+00$_{5.35e-01}$[2]	1.670e+02$_{3.32e+01}$[6]	1.514e+01$_{1.84e+00}$[4]	3.808e+01$_{1.62e+00}$[5]	1.404e+01$_{1.96e+00}$[3]
WFG7	1.362e+00$_{1.05e-01}$[1]	4.799e+02$_{6.39e+01}$[7]	7.428e+00$_{7.07e-01}$[2]	1.748e+02$_{2.53e+01}$[6]	1.663e+01$_{2.02e+00}$[4]	4.790e+01$_{1.35e+00}$[5]	1.403e+01$_{1.77e+00}$[3]
WFG8	1.385e+00$_{1.43e-01}$[1]	2.570e+02$_{3.33e+01}$[7]	6.980e+00$_{8.11e-01}$[2]	1.563e+02$_{1.92e+01}$[6]	2.076e+01$_{2.47e+00}$[4]	3.445e+01$_{1.32e+00}$[5]	1.832e+01$_{2.76e+00}$[3]
WFG9	1.579e+00$_{1.93e-01}$[1]	5.459e+02$_{7.49e+01}$[7]	7.239e+00$_{1.06e+00}$[2]	1.630e+02$_{2.26e+01}$[6]	1.703e+01$_{2.23e+00}$[4]	5.107e+01$_{2.78e+00}$[5]	1.468e+01$_{2.00e+00}$[3]
RE2-4-1	1.470e+00$_{3.69e-01}$[1]	1.393e+01$_{2.37e+00}$[4]	1.531e+01$_{1.98e+00}$[6]	7.948e+00$_{1.19e+00}$[2]	1.427e+01$_{1.76e+00}$[5]	1.996e+01$_{8.88e-01}$[7]	1.236e+01$_{1.49e+00}$[3]
RE2-3-2	1.085e+00$_{1.78e-01}$[1]	1.245e+01$_{1.44e+00}$[4]	1.363e+01$_{1.49e+00}$[5]	7.216e+00$_{8.25e-01}$[2]	1.408e+01$_{1.68e+00}$[6]	1.961e+01$_{8.68e-01}$[7]	1.197e+01$_{1.30e+00}$[3]
RE2-4-3	1.534e+00$_{4.44e-01}$[1]	1.149e+01$_{1.31e+00}$[5]	6.491e+00$_{1.32e+00}$[2]	8.369e+00$_{1.13e+00}$[3]	1.347e+01$_{1.69e+00}$[6]	1.812e+01$_{1.41e+00}$[7]	1.099e+01$_{1.41e+00}$[4]
RE2-2-4	1.686e+00$_{5.77e-01}$[1]	6.149e+02$_{5.45e+01}$[7]	1.883e+01$_{2.49e+00}$[4]	3.385e+02$_{3.86e+01}$[6]	1.613e+01$_{1.93e+00}$[3]	3.120e+01$_{2.61e+00}$[5]	1.410e+01$_{1.59e+00}$[2]
RE2-3-5	1.216e+00$_{2.72e-01}$[1]	1.189e+01$_{1.23e+00}$[4]	9.953e+00$_{7.43e-01}$[3]	7.673e+00$_{8.03e-01}$[2]	1.404e+01$_{1.29e+00}$[6]	2.059e+01$_{6.55e-01}$[7]	1.193e+01$_{1.01e+00}$[5]
RE3-3-1	9.999e−01$_{1.45e-01}$[1]	1.177e+01$_{1.13e+00}$[4]	9.889e+00$_{7.25e-01}$[3]	7.782e+00$_{8.19e-01}$[2]	1.397e+01$_{1.41e+00}$[6]	1.998e+01$_{6.11e-01}$[7]	1.241e+01$_{1.28e+00}$[5]
RE3-4-2	1.119e+00$_{4.87e-02}$[1]	1.140e+01$_{1.09e+00}$[5]	3.501e+00$_{4.67e-01}$[2]	8.042e+00$_{7.24e-01}$[3]	1.331e+01$_{1.32e+00}$[6]	1.846e+01$_{7.21e-01}$[7]	1.107e+01$_{1.02e+00}$[4]
Mean Rank	1.000	5.545	2.500	5.091	4.636	5.818	3.409

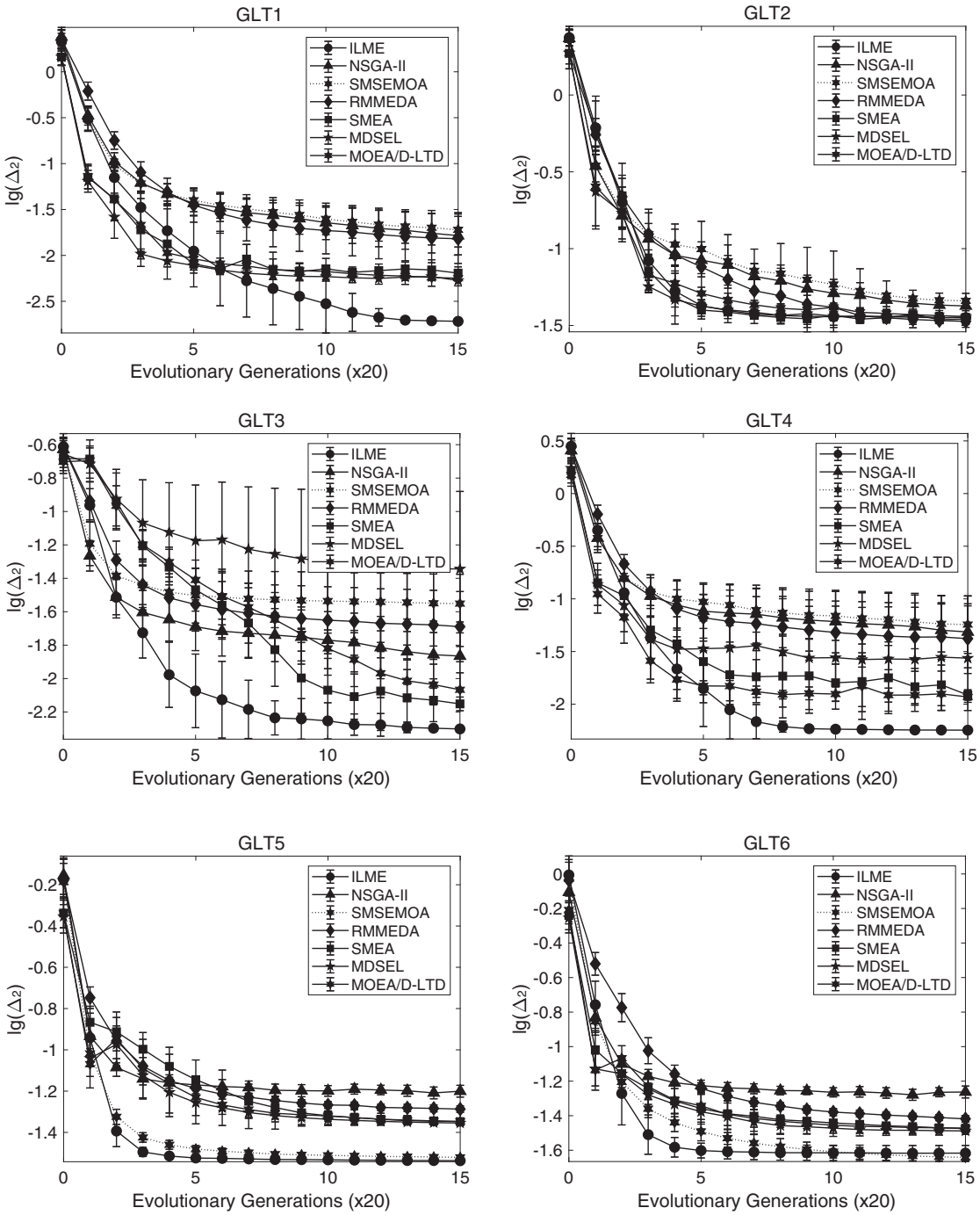


Fig. 4. Evolution of the mean and *sd* of Δ_2 obtained by six algorithms over 31 independent runs on the GLT test suites.

Table 2 shows the statistics for the Δ_2 metric for the seven algorithms. As can be seen from the table, ILME has significantly better Δ_2 values than NSGA-II, SMS-EMOA, RM-MEDA, SMEA, MDSEL, or MOEA/D-LTD for 17, 17, 17, 11, 16, and 21 out of the 22 test instances, respectively. In terms of the overall mean rankings, ILME is best, followed by SMEA, MDSEL, RM-MEDA, SMS-EMOA, MOEA/D-LTD, and NSGA-II. ILME was remarkably better than the other algorithms on the GLT and WFG test suites, except for WFG1 and WFG5, on which ILME had average performance. For the RE problems, ILME was slightly inferior to MDSEL on the RE3-3–1 and RE3-3–1 problems. In summary, ILME is superior to the other algorithms on the GLT and WFG test suites.

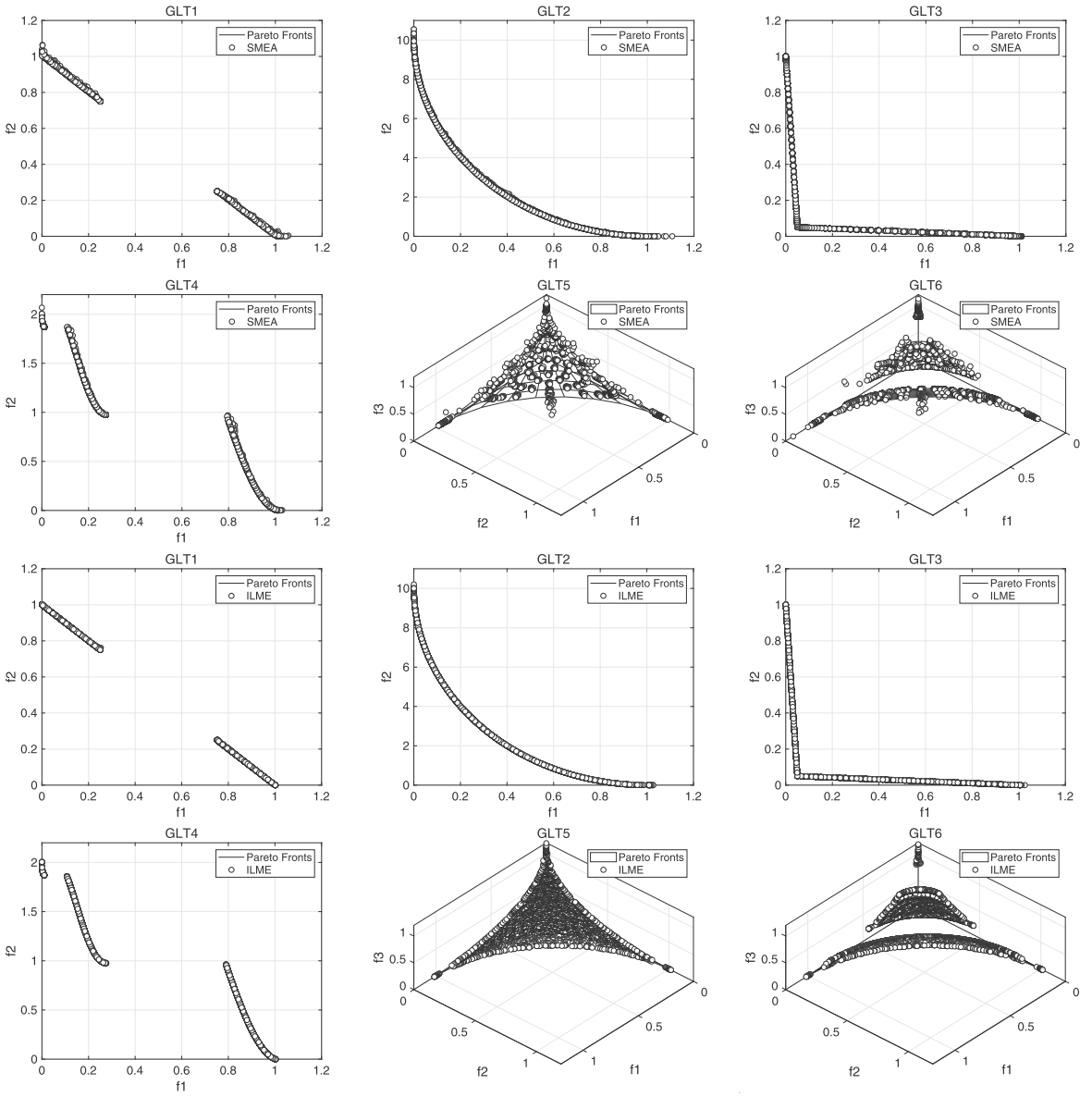


Fig. 5. Overall final populations obtained by SMEA and ILME on the GLT test suites with 31 independent runs.

Table 3 lists the experimental results for the HV values. It shows that ILME was significantly better than NSGA-II, SMS-EMOA, RM-MEDA, SMEA, MDSEL, and MOEA/D-LTD for 17, 19, 17, 14, 15, and 21 out of the 22 test instances, respectively. ILME was in first place, followed by MDSEL, SMEA, RM-MEDA, SMS-EMOA, MOEA/D-LTD, and NSGA-II, in terms of the overall mean rankings. ILME was significantly superior on the GLT and WFG test suites, except for WFG1 and WFG5. For the RE problems, ILME was best for 4, 5, 4, 4, 2, and 7 of 7 tests, worse for 2, 1, 1, 3, 7, and 0 tests, and similar for 1, 1, 2, 0, 1, and 0 tests, respectively, than NSGA-II, SMS-EMOA, RM-MEDA, SMEA, MDSEL, and MOEA/D-Ltd. ILME works particularly well on the GLT and WFG test suites, on average, in terms of HV.

Table 4 summarizes the run times of the seven algorithms on the 22 test instances averaged over 31 independent runs. As can be seen from the table, NSGA-II consumes the least time, whereas the run times increase in order for RM-MEDA, ILME, MDSEL, SMEA, SMS-EMOA, and MOEA/D-Ltd. The run time for NSGA-II was always significantly shorter than those of the other algorithms, simply because NSGA-II has a much lower time complexity, especially in the best case. Even the worst-case time complexity of NSGA-II is equal to or less than that of the other algorithms in their best cases. On the other hand, though the complexity of ILME is higher than that of SMEA, MDSEL, and MOEA/D-LTD in the worst case, ILME almost always spends less time than these three algorithms, especially for the three-objective problems (GLT5 and GLT6). This implies that the learning mechanism of ILME is more effective than those for the other learning-based algorithms.

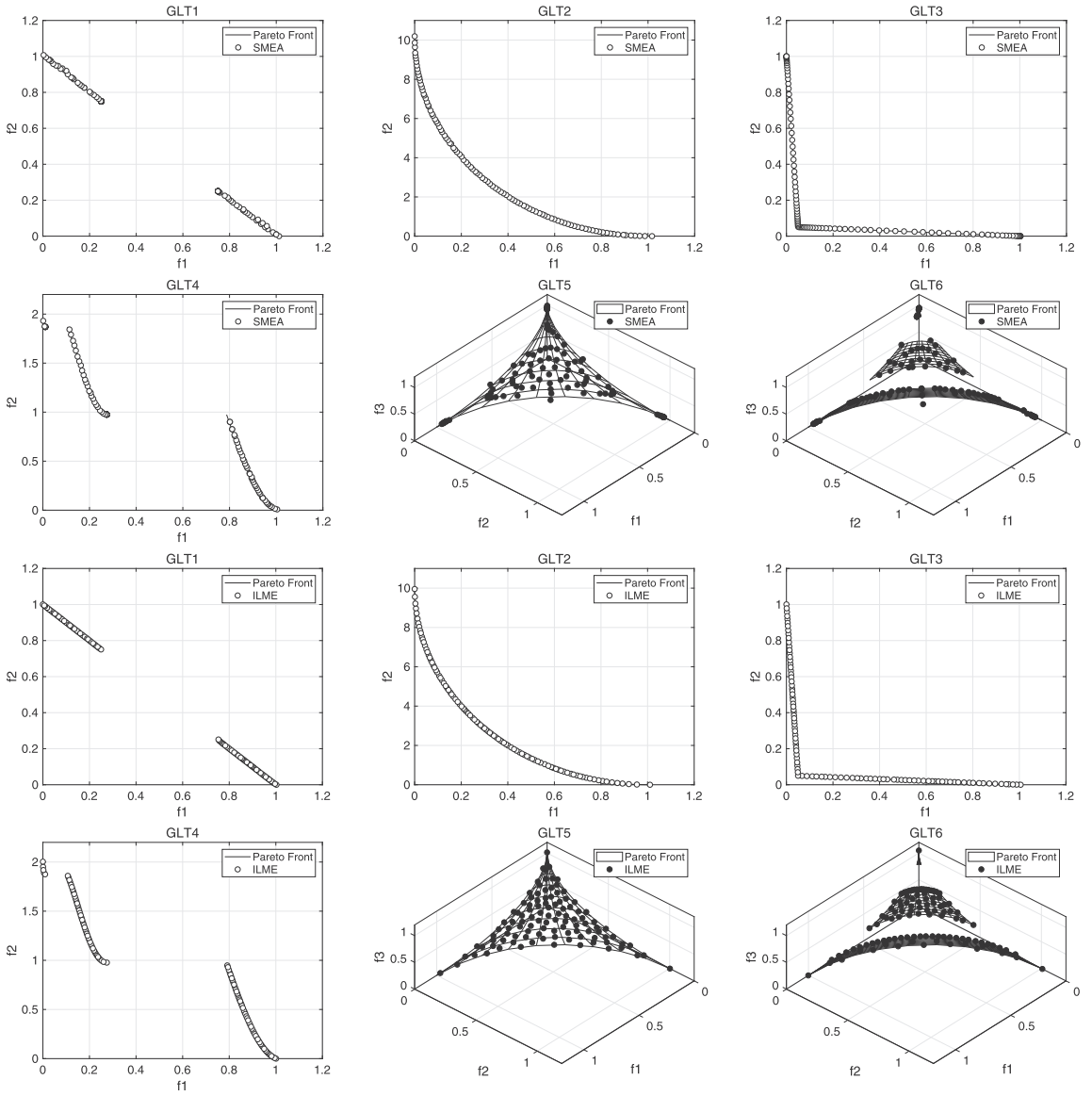


Fig. 6. Representative final populations obtained by SMEA and ILME on the GLT test suites with 31 independent runs.

To investigate the convergence speed, Fig. 4 shows the mean and *sd* of Δ_2 versus the number of evolutionary generations for the seven algorithms for 31 independent runs of the GLT test suites. As can be seen from the figure, ILME has the lowest mean Δ_2 values after 300 evolutionary generations for GLT1 and GLT3–GLT5, which confirms that ILME has the overall fastest convergence of the algorithms for the GLT test suites. More specifically, the search efficiency of ILME is superior to that of all the other algorithms on GLT3, GLT5, and GLT6 throughout the evolutionary process. ILME has an average convergence speed early in the evolutionary process for GLT1 and GLT4, when it is particularly inferior to MDSEL and MOEA/D-Ltd. However, ILME converges steadily, whereas the convergence of MDSEL and MOEA/D-LTD slows down. In addition, the search efficiency of ILME is similar to that of SMEA and MDSEL on GLT2, whereas the search efficiency of ILME is close to SMS-EMOA on GLT5.

To demonstrate the superiority of ILME further, we visualize the data for ILME and SMEA, which are the best two algorithms in terms of the mean ranking in Table 2. The data are for 31 independent runs on the GLT1–GLT6 test instances. The overall final populations are illustrated in Fig. 5, whereas Fig. 6 shows representative final populations with median Δ_2 values. As can be seen from Fig. 5, the final populations of ILME have converged to the PFs and there is a uniform distribution on the PFs for each test instance. In contrast, SMEA either did not converge to the PFs or the distribution is rather non-uniform. For GLT5 and GLT6, the performance of ILME was obviously superior to SMEA, and the performance of SMEA was inferior to

Table 5
Mean and *sd* of Δ_2 and HV metric after 31 independent runs of four algorithms on the GLT test suites.

Instance	ILME-SCSS	ILME-RSS	ILME-DE	ILME
		Δ_2		
GLT1	$2.574e{-}03^{\dagger}_{4.06e-04}$[2]	$2.657e{-}03^{\dagger}_{3.97e-04}$[3]	$2.818e{-}03^{\dagger}_{2.21e-04}$[4]	$1.915e{-}03_{5.47e-05}$[1]
GLT2	$3.538e{-}02^{\S}_{3.14e-03}$[1]	$3.605e{-}02^{\dagger}_{5.48e-03}$[3]	$3.651e{-}02^{\dagger}_{1.20e-03}$[4]	$3.556e{-}02_{1.13e-03}$[2]
GLT3	$6.950e{-}03^{\dagger}_{5.01e-04}$[2]	$4.484e{-}02^{\dagger}_{8.56e-02}$[4]	$8.484e{-}03^{\dagger}_{2.24e-03}$[3]	$4.993e{-}03_{8.26e-05}$[1]
GLT4	$1.215e{-}02^{\dagger}_{1.31e-03}$[3]	$2.712e{-}02^{\dagger}_{5.85e-02}$[4]	$1.156e{-}02^{\dagger}_{1.43e-03}$[2]	$5.665e{-}03_{6.87e-05}$[1]
GLT5	$3.953e{-}02^{\dagger}_{1.05e-03}$[3]	$3.890e{-}02^{\dagger}_{7.92e-04}$[2]	$3.958e{-}02^{\dagger}_{1.10e-03}$[4]	$2.902e{-}02_{3.26e-04}$[1]
GLT6	$2.591e{-}02^{\dagger}_{2.25e-03}$[2]	$2.598e{-}02^{\dagger}_{6.36e-04}$[3]	$2.680e{-}02^{\dagger}_{7.61e-04}$[4]	$2.416e{-}02_{2.61e-03}$[1]
		HV		
GLT1	$3.369e{+}00^{\dagger}_{2.01e-03}$[4]	$3.370e{+}00^{\dagger}_{2.07e-03}$[3]	$3.370e{+}00^{\dagger}_{2.28e-03}$[2]	$3.371e{+}00_{1.35e-03}$[1]
GLT2	$1.981e{+}01^{\dagger}_{1.21e-02}$[2]	$1.980e{+}01^{\dagger}_{2.50e-02}$[3]	$1.979e{+}01^{\dagger}_{4.88e-03}$[4]	$1.981e{+}01_{7.32e-04}$[1]
GLT3	$3.948e{+}00^{\dagger}_{3.33e-04}$[2]	$3.936e{+}00^{\dagger}_{2.87e-02}$[4]	$3.947e{+}00^{\dagger}_{4.19e-04}$[3]	$3.949e{+}00_{9.22e-05}$[1]
GLT4	$4.987e{+}00^{\dagger}_{3.84e-03}$[3]	$4.943e{+}00^{\dagger}_{1.79e-01}$[4]	$4.988e{+}00^{\dagger}_{3.03e-03}$[2]	$4.993e{+}00_{3.70e-04}$[1]
GLT5	$7.965e{+}00^{\dagger}_{1.48e-03}$[3]	$7.966e{+}00^{\dagger}_{7.12e-04}$[2]	$7.964e{+}00^{\dagger}_{8.57e-04}$[4]	$7.969e{+}00_{1.01e-04}$[1]
GLT6	$7.957e{+}00^{\dagger}_{2.89e-03}$[4]	$7.960e{+}00^{\dagger}_{1.14e-03}$[2]	$7.958e{+}00^{\dagger}_{7.69e-04}$[3]	$7.961e{+}00_{2.46e-03}$[1]
Mean Rank	2.583	3.083	3.250	1.083
†/§/≈	11/1/0	12/0/0	12/0/0	

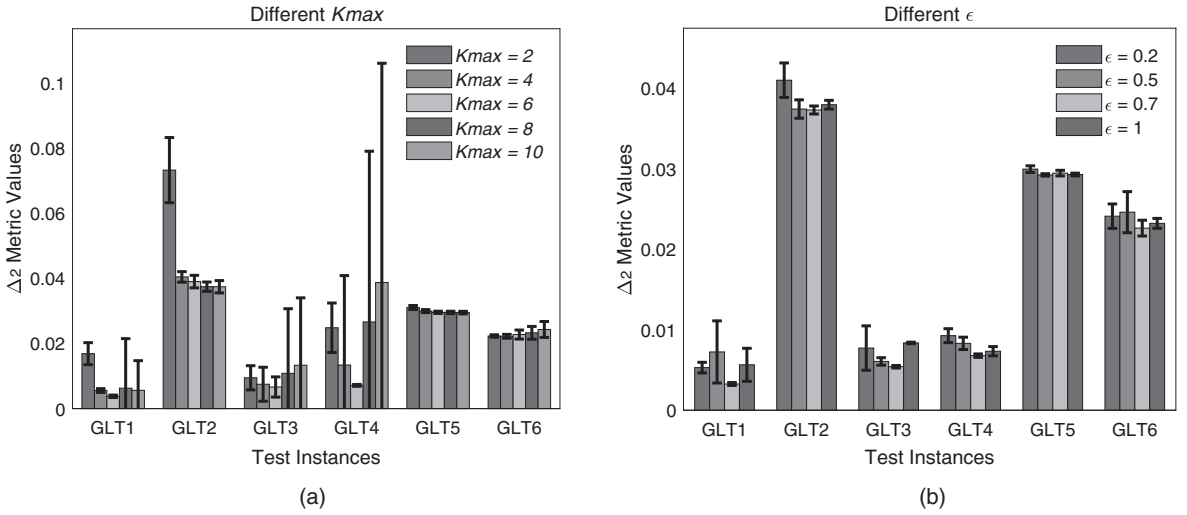


Fig. 7. Mean Δ_2 metric values with 31 runs versus of K_{max} in ILME for GLT test suites.

ILME for the remaining test instances. Moreover, the representative final populations obtained by the two algorithms are good approximations of the PFs for each instance, although some segments for GLT3 and GLT4 are missing for SMEA. The objective points yielded by ILME in the representative final populations are uniformly placed across the PFs on GLT1–GLT6, whereas SMEA failed to provide a uniform distribution of points on the PFs for GLT5 and GLT6. In summary, the approximate fronts achieved by ILME on the GLT test suites have excellent convergence and outstanding diversity.

5. Further discussion

5.1. Effectiveness of the proposed sampling strategy

SCSS and DE were hybridized in ILME to produce new offspring. To validate the effectiveness of the proposed sampling strategy, three versions of the algorithm (ILME-SCSS, ILME-RSS, and ILME-DE) were compared with ILME. For ILME-SCSS (ILME-RSS and ILME-DE), all the components and settings were the same as for ILME, except that SCSS (RSS and DE) was used to generate solutions. The Δ_2 and HV values of the final objective populations obtained by ILME-SCSS, ILME-RSS, ILME-DE, and ILME on the GLT test suites are summarized in Table 5. As can be seen, ILME has 11 out of 12 the best metric values and the top mean rank, which shows that ILME is significantly better than the other versions. To be specific, ILME-SCSS performs better than ILME-RSS, which benefits from its powerful search capability for exploration. The experimental results show that the sampling strategy developed dramatically enhances the performance of ILME.

5.2. Sensitivity analysis on algorithmic parameters

5.2.1. Parameter sensitivity analysis of K_{max}

K_{max} is an important parameter in ILME. It is the maximum number of components in the learning model used to discover the manifold structure of the PS. To study the sensitivity of this parameter, ILME was run on the GLT test suites for $K_{max} \in \{2, 4, 6, 8, 10\}$. As K_{max} increases, the model to learn the manifold structure of PS becomes more accurate. All the other parameters were the same as in Section 4.3. For each configuration, 31 independent runs were conducted for each instance.

As can be seen from Fig. 7a, the performance of ILME is very robust for different K_{max} values on the GLT2, GLT5, and GLT6 test instances. However, changes to K_{max} lead to relatively large performance differences for GLT1, GLT3, and GLT4. In particular, the performance of ILME is average when K_{max} is too large or too small. In general, a medium K_{max} value in ILME could result in excellent experimental results with the GLT test suite.

5.2.2. Parameter sensitivity analysis of ε

In ILME, the exploitation probability ε balances the effectiveness of exploitation and exploration. SCSS is used for exploitation with a probability ε, otherwise the DE operator is applied for exploration. ILME was run on the GLT test suites for $\varepsilon \in \{0.2, 0.5, 0.7, 1\}$. As ε increases, the solution generation method becomes more inclined to use SCSS instead of the DE operator. All the other parameters were the same as in Section 4.3. For each configuration, 31 independent runs were conducted on each instance.

As shown in Fig. 7b, different values of ε have different effects on the performance of ILME for the GLT1–GLT6 test instances. Overall, the performance of ILME is not sensitive to the exploitation probability ε. Remarkably, ILME performs better with larger ε values. Its convergence efficiency for the GLT test suites is best when $\varepsilon = 0.7$. Note, though, that the optimal value of ε depends on the problem.

5.2.3. Parameter sensitivity analysis of CR and F

In ILME, the crossover constant CR controls the effect of the parent on the offspring solutions. The higher CR is, the less effect it has. The differentiation constant F scales the influence of the set of pairs of solutions selected to calculate the mutation value.

We investigated how ILME is influenced by different combinations of the two parameters for GLT1–GLT4, since ILME is not sensitive to the values of these parameters on GLT5 and GLT6, according to our numerical experiments. We tested 16 combinations with four values of CR $\in \{0.2, 0.5, 0.7, 1\}$ and four values of $F \in \{0.2, 0.5, 0.7, 1\}$. All the other parameters were the same as in Section 4.3. For each configuration, 31 independent runs were conducted on each instance.

Fig. 8 shows the mean Δ_2 values for different combination of CR and F on the selected test instances. ILME performed best when both CR and F were large (such as CR = 0.5, 0.7, or 1 and F = 0.5, 0.7, or 1). There were slight fluctuations in performance for the GLT1 and GLT4 test instances. These results demonstrate that large values of CR and F are better. The combination of CR $= 1$ and $F = 0.7$ is good for the majority of test instances.

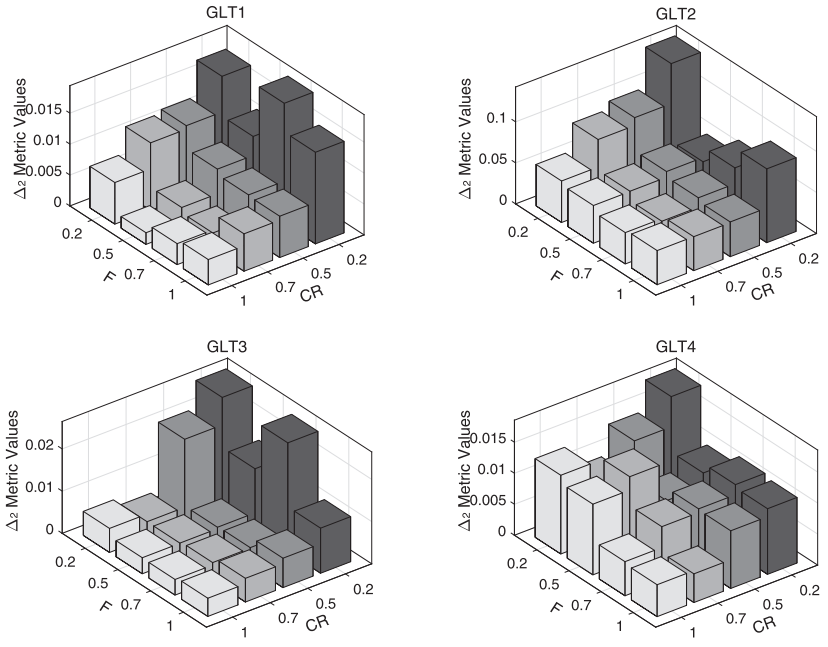


Fig. 8. Mean Δ_2 metric values with 31 runs versus of CR and F in ILME for GLT1–GLT4 test suites.

6. Conclusions

In this paper, an incremental-learning model-based multiobjective estimation of distribution algorithm, termed ILME, has been proposed for solving MOPs. The incremental-learning model helps to reduce the computational cost of learning knowledge from the properties of the PS. In ILME, each new solution generated during the evolution becomes a datum in the evolving data stream, which is fed incrementally into the learning model to capture adaptively the structure of the PS. The parameters of the model are updated continually as newly generated data are collected. Each datum is learned only once for the model, regardless of whether it has been preserved or deleted. Moreover, a sampling strategy based on the learned model has been designed to balance the exploration/exploitation dilemma in evolutionary searches.

Numerical experiments have been performed to compare the performance of ILME with six representative MOEAs (NSGA-II, SMS-EMOA, RM-MEDA, SMEA, MDSEL, and MOEA/D-LTD) for 22 test instances with complex PF or complicated PS structures. This systematic study has demonstrated that ILME outperforms, or is competitive with, the other algorithms in terms of convergence and diversity, as measured by Δ_2 and HV. ILME has an average performance on RE problems. Moreover, the effectiveness of the proposed sampling strategy and the sensitivity of the algorithmic parameters have been experimentally investigated.

Future work might involve the implementation of better strategies to reduce the computational costs of MOEA. A potentially important issue in future research is the development of more advanced incremental-learning mechanisms, since the combination of current learning mechanisms and the evolutionary search is inadequate. Moreover, in future work, we will pay more attention to real-world problems [43] and many-objective optimization.

CRediT authorship contribution statement

Tingrui Liu: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Visualization. **Xin Li:** Writing - review & editing, Supervision. **Liguo Tan:** Project administration, Funding acquisition. **Shenmin Song:** Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to express sincere appreciation to constructive and valuable comments from anonymous reviewers. This study was funded by the Science Fund for Excellent Young Scholars of Heilongjiang Province (Grant No.: YQ2020F007), National Natural Science Foundation of China (Grant No.: 6191101340), the Defense Industrial Technology Development Program.

References

[1] S. Rostami, F. Neri, M. Epitropakis, Progressive preference articulation for decision making in multi-objective optimisation problems, Integrated Computer-Aided Engineering 24 (4) (2017) 315–335.
[2] M. Wu, K. Li, S. Kwong, Q. Zhang, J. Zhang, Learning to decompose: a paradigm for decomposition-based multiobjective optimization, IEEE Transactions on Evolutionary Computation 23 (3) (2019) 376–390.
[3] Q. Zhang, A. Zhou, Y. Jin, RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 41–63.
[4] A. Zhou, Q. Zhang, G. Zhang, A multiobjective evolutionary algorithm based on decomposition and probability model, in: 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–8.
[5] X. Ma, F. Liu, Y. Qi, L. Li, L. Jiao, M. Liu, J. Wu, MOEA/D with Baldwinian learning inspired by the regularity property of continuous multiobjective problem, Neurocomputing 145 (2014) 336–352.
[6] H. Zhang, X. Zhang, X.Z. Gao, S. Song, Self-organizing multiobjective optimization based on decomposition with neighborhood ensemble, Neurocomputing 173 (2016) 1868–1884.
[7] X. Li, H. Zhang, S. Song, A self-adaptive mating restriction strategy based on survival length for evolutionary multiobjective optimization, Swarm and Evolutionary Computation 43 (2018) 31–49.
[8] H. Zhang, J. Sun, T. Liu, K. Zhang, Q. Zhang, Balancing exploration and exploitation in multiobjective evolutionary optimization, Information Sciences 497 (2019) 129–148.
[9] Y. Liu, H. Ishibuchi, N. Masuyama, Y. Nojima, Adapting reference vectors and scalarizing functions by growing neural gas to handle irregular pareto fronts, IEEE Transactions on Evolutionary Computation 24 (3) (2020) 439–453.
[10] J. Sun, H. Zhang, A. Zhou, Q. Zhang, K. Zhang, Z. Tu, K. Ye, Learning from a stream of nonstationary and dependent data in multiobjective evolutionary optimization, IEEE Transactions on Evolutionary Computation 23 (4) (2019) 541–555.
[11] H. Zhang, A. Zhou, S. Song, Q. Zhang, X.Z. Gao, J. Zhang, A self-organizing multiobjective evolutionary algorithm, IEEE Transactions on Evolutionary Computation 20 (5) (2016) 792–806.
[12] T. Liu, X. Li, L. Tan, S. Song, A novel adaptive greedy strategy based on gaussian mixture clustering for multiobjective optimization, Swarm and Evolutionary Computation 61 (2021) 100815.
[13] R. Cheng, Y. Jin, K. Narukawa, B. Sendhoff, A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling, IEEE Transactions on Evolutionary Computation 19 (6) (2015) 838–856.
[14] K. Li, S. Kwong, A general framework for evolutionary multiobjective optimization via manifold learning, Neurocomputing 146 (2014) 65–74.

[15] S. Calinon, A. Billard, Incremental learning of gestures by imitation in a humanoid robot, in: HRI 2007 – Proceedings of the 2007 ACM/IEEE Conference on Human-Robot Interaction – Robot as Team Member, 2007, pp. 255–262..
[16] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[17] D. Corne, N. Jerram, J. Knowles, M. Oates, J. Martin, PESA-II: region-based selection in evolutionary multiobjective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2001, pp. 283–290.
[18] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach. Part I: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601.
[19] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.
[20] D.H. Phan, J. Suzuki, R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization, in: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, IEEE, 2013, pp. 1836–1845..
[21] C.A. Rodríguez Villalobos, C.A. Coello Coello, A new multi-objective evolutionary algorithm based on a performance assessment indicator, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO '12, Association for Computing Machinery, New York, USA, 2012, pp. 505–512.
[22] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.
[23] H.L. Liu, F.Q. Gu, Y.M. Cheung, T-MOEA/D: MOEA/D with objective transform in multi-objective problems, in: Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010, Vol. 2, 2010, pp. 282–285..
[24] M. Laumanns, J. Ocenasek, Bayesian optimization algorithms for multi-objective optimization, Lecture Notes in Computer Science 2439 (2002) 298–307.
[25] M. Pelikan, K. Sastry, D.E. Goldberg, Multiobjective hBOA, clustering, and scalability, in: GECCO 2005 – Genetic and Evolutionary Computation Conference, 2005, pp. 663–670..
[26] C.W. Ahn, R.S. Ramakrishna, Multiobjective real-coded bayesian optimization algorithmrevisited: Diversity preservation, in: Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference, ACM Press, 2007, pp. 593–600..
[27] M.S. Martins, M.R. Delgado, R. Santana, R. Lüders, R. Aderbal, C.P. De Almeida, HMOBEDA: Hybrid multi-objective Bayesian Estimation of distribution algorithm, in: GECCO 2016 – Proceedings of the 2016 Genetic and Evolutionary Computation Conference, ACM Press, 2016, pp. 357–364.
[28] M.S.R. Martins, M. Delgado, R. Luders, R. Santana, R.A. Goncalves, C.P. De Almeida, Probabilistic analysis of pareto front approximation for a hybrid multi-objective Bayesian estimation of distribution algorithm, in: Proceedings – 2017 Brazilian Conference on Intelligent Systems, IEEE, 2017, pp. 384–389..
[29] E.C. Garrido-Merchan, D. Hernandez-Lobato, Predictive entropy search for multi-objective bayesian optimization with constraints, Neurocomputing 361 (Oct. 7) (2019) 50–68..
[30] P.A. Bosman, D. Thierens, Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms, International Journal of Approximate Reasoning 31 (3) (2002) 259–289.
[31] H. Li, Q. Zhang, E. Tsang, J.A. Ford, Hybrid estimation of distribution algorithm for multiobjective knapsack problem, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2004, pp. 145–154.
[32] P.A. Bosman, D. Thierens, Adaptive variance scaling in continuous multi-objective estimation-of-distribution algorithms, in: Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference, 2007, pp. 500–507..
[33] P.A. Bosman, The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization, in: Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10, 2010, pp. 351–358..
[34] V.A. Shim, K.C. Tan, C.Y. Cheong, J.Y. Chia, Enhancing the scalability of multi-objective optimization via restricted Boltzmann machine-based estimation of distribution algorithm, Information Sciences 248 (2013) 191–213.
[35] E. Mohagheghi, M.R. Akbarzadeh, Multi-objective Estimation of Distribution Algorithm based on Voronoi and local search, in: 2016 6th International Conference on Computer and Knowledge Engineering, ICCKE 2016, IEEE, 2016, pp. 54–59..
[36] S. Maza, M. Touahria, Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms, Applied Intelligence 49 (12) (2019) 4237–4257.
[37] Q. Yuan, G. Dai, Y. Zhang, A novel multi-objective evolutionary algorithm based on LLE manifold learning, Engineering with Computers 33 (2) (2017) 293–305.
[38] C. He, S. Huang, R. Cheng, K.C. Tan, Y. Jin, Evolutionary multiobjective optimization driven by generative adversarial networks (gans), IEEE Transactions on Cybernetics (2020) 1–14.
[39] X. Li, H. Zhang, S. Song, MOEA/D with the online agglomerative clustering based self-adaptive mating restriction strategy, Neurocomputing 339 (2019) 77–93.
[40] R. Xu, D.C. Wunsch, Clustering, vol. 10, John Wiley & Sons, 2008.
[41] F. Gu, H.L. Liu, K.C. Tan, A multiobjective evolutionary algorithm using dynamic weight design method, International Journal of Innovative Computing, Information and Control 8 (5 B) (2012) 3677–3688..
[42] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Transactions on Evolutionary Computation 10 (5) (2006) 477–506.
[43] R. Tanabe, H. Ishibuchi, An easy-to-use real-world multi-objective optimization problem suite, Applied Soft Computing 89..
[44] S. Rostami, F. Neri, Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm, Integrated Computer-Aided Engineering 23 (4) (2016) 313–329.
[45] S. Rostami, F. Neri, A fast hypervolume driven selection mechanism for many-objective optimisation problems, Swarm and Evolutionary Computation 34 (2017) 50–67.
[46] S. Rostami, F. Neri, K. Gyaurski, On algorithmic descriptions and software implementations for multi-objective optimisation: A comparative study, SN Computer Science 1 (5) (2020) 1–23.
[47] Y. Tian, R. Cheng, X. Zhang, Y. Jin, Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum], IEEE Computational Intelligence Magazine 12 (4) (2017) 73–87.
[48] K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Computer Science and Informatics 26 (1) (1996) 30–45.