Soft Computing
https://doi.org/10.1007/s00500-018-3390-8

FOUNDATIONS

A latent space-based estimation of distribution algorithm for large-scale global optimization

Wenyong Dong[1,2] · **Yufeng Wang**[1,2] · **Mengchu Zhou**[3]

Abstract
Large-scale global optimization problems (LSGOs) have received considerable attention in the field of meta-heuristic algorithms. Estimation of distribution algorithms (EDAs) are a major branch of meta-heuristic algorithms. However, how to effectively build the probabilistic model for EDA in high dimensions is confronted with obstacle, making them less attractive due to the large computational requirements. To overcome the shortcomings of EDAs, this paper proposed a latent space-based EDA (LS-EDA), which transforms the multivariate probabilistic model of Gaussian-based EDA into its principal component latent subspace with lower dimensionality. LS-EDA can efficiently reduce the complexity of EDA while maintaining its probability model without losing key information to scale up its performance for LSGOs. When the original dimensions are projected to the latent subspace, those dimensions with larger projected value make more contribution to the optimization process. LS-EDA can also help recognize and understand the problem structure, especially for black-box optimization problems. Due to dimensionality reduction, its computational budget and population size can be effectively reduced while its performance is highly competitive in comparison with the state-of-the-art meta-heuristic algorithms for LSGOs. In order to understand the strengths and weaknesses of LS-EDA, we have carried out extensive computational studies. Our results revealed LS-EDA outperforms the others on the benchmark functions with overlap and nonseparate variables.

Keywords Probabilistic PCA · Estimation of distribution algorithm (EDA) · Maximum likelihood estimate (MLE)

1 Introduction

Meta-heuristic algorithms, such as evolution-based algorithms (EAs) (Fogel 1996), physics-based algorithms (PAs) and swarm intelligence-based algorithms (SIAs) (Yang 2014), have been successfully applied to a great deal of optimization problems with low dimension. However, they have faced with many challenges when they are used to solve some real-world applications. Many real-world applications, such as large-scale and super-large-scale circuit design (Lohn and Colombano 1999), intelligent transportation planning (Che et al. 2015), economic or social system optimization (Ray and Liew 2003) and inverse problems in biological systems (Wu et al. 2012), often are related to large-scale global optimization problems (LSGOs), which require optimizing a mass of parameters. A well-known example is the S system model from biological systems in that $2N(N+1)$ parameters need to be optimized and N tends to exceed 1000, where N denotes the number of components (Mahdavi et al. 2015). Again, their parameters generally are mutually dependent and interactive to make the final decision.

The difficulty of optimizing LSGOs mainly stems from two aspects as follows. The first one is the curse of dimension which means the search space exponentially grows as the number of dimensions increases. The other one is a large number of dimensions leading to the emergence of complex behaviors and characteristics in its landscape, such as multimodal and singularity. For example, the Rosenbrock

Communicated by A. Di Nola.

✉ Yufeng Wang
wangyufeng@whu.edu.cn

Wenyong Dong
dwy@whu.edu.cn

Mengchu Zhou
mengchu.zhou@njit.edu

[1] Computer School, Wuhan University, Wuhan 430072, China

[2] Software School, Nanyang Institute of Technology, Nanyang 473000, Henan, China

[3] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA

Published online: 13 July 2018

function is a unimodal function in two dimensions, but it turns into a multimodal function when the number of dimensions increases (Shang and Qiu 2006).

Up to now, several particular mechanisms have been proposed to solve LSGOs. Mahdavi et al. (2015) summarize them into two main categories: decomposition (or grouping)-based cooperative coevolution (CC) strategies (Mahdavi et al. 2016a) and nondecomposition-based algorithms. The former belongs to a divide-and-conquer strategy. It first divides whole problem into several subcomponents with low dimensions and then cooperatively optimizes them by using different meta-heuristic algorithms. The latter resorts to specific effective operators or hybridizes different algorithms to straightforwardly solve LSGOs as a whole.

Although grouping-based strategies have made significant advances in solving LSGOs, some prominent shortcomings remain to be overcome. The interactions among variables are not considered in random grouping and fixed grouping strategies. Even though the dependencies among variables are detected in differential grouping (DG) (Omidvar et al. 2014), its grouping performance may degenerate with more and more dimensions since DG seriously relies on the setting of its differential parameter ε. Therefore, new methods are worthy to be designed to solve LSGOs.

EDAs (Mühlenbein and Paass 1996), known as an important branch of meta-heuristic algorithms, have been intensively studied for global optimization for two decades. They were initially used to solve combinatorial optimization problems and then extended to continuous function optimization. EDAs can extract some underlying interactions among the variables through their learned models. Santana et al. (2013) introduced a network theory for mining structural information for evolutionary optimization. Hauschild et al. (2008) proposed two different methods to restrict or penalize the edges in the hierarchical Bayesian optimization algorithm (hBOA) model building: (a) computation of the probability coincidence matrix and (b) computation of a distance-based metric among variables from a known problem structure. Santana et al. (2012) adopted a Bayesian network to transfer the structural information to bias the construction of an aggregation matrix in a multi-marker tagging single nucleotide polymorphism (SNP) selection problem. Armananzas et al. (2011) selected peak in mass spectrometry data by using an ensemble approach that integrates information from different probabilistic models learned by an EDA.

EDAs have been successfully applied to many low-dimensional optimization problems. However, as space dimensionality increases, their performance declines very quickly. Some new variants of EDAs are proposed to overcome this problem. Pok (2005) adopted independent component analysis (ICA) to render a recombination or crossover operator independent of the coordinate system. Mishra and Gallagher (2014) proposed a new screening estimation of distribution algorithm (sEDA) that utilizes the objective function values obtained during the search. According to the estimated sensitivity of the fitness function to individual variables in the search space, the rank of the covariance matrix in the high-dimensional problem is thus reduced. Kabn et al. (2015) and Sanyang and Kabn (2015) proposed a method called random projections estimation of distribution algorithm (RP-EDA). It introduces an ensemble of random projections (RP) to compress the full covariance. Although the mentioned EDA variants have low computational complexity, they cannot capture the strong interdependencies among variables, and may show poor performance in solving complex problems (Mahdavi et al. 2016b). Dimensionality reduction technology is a traditional strategy in pattern recognition for high-dimensionality case, but is still not be introduced into the optimization algorithms to solve LSGOs. We conclude that dimensionality reduction for a probabilistic model of EDA can efficiently reduce the complexity of continuous EDA while maintaining its probability model without losing key information to scale up its performance for LSGOs.

To reduce the computational burden and speed up convergence, this work proposes a latent space-based estimation of distribution algorithm (LS-EDA), which changes the original searching models into low-dimensional manifold. LS-EDA uses an iterative expectation–maximization (EM) algorithm to find the solution of maximum likelihood for probability models with latent variables instead of eigenvalue decomposition to reduce computing time. The analysis for the proposed LS-EDA shows that it possess the merits resulting from both univariate EDAs and multivariate Gaussian EDAs. We aim to address the issue of improving the performance of EDA facing the increasing dimensionality of a search space. In a high-dimensional search space, we can use the latent variables to estimate the probabilistic model and visually analyze the problem structure by observing the obtained probabilistic model. In every sampling step, we search the best solution along the direction of the principal subspace. Due to dimensionality reduction, the computational budget and the population size can be greatly reduced in LS-EDA.

The remainder of this paper is organized as follows. In Sect. 2, we review background knowledge including EDA, probabilistic principal component analysis (pPCA) and expectation–maximization (EM) algorithm. In Sect. 3, we present the proposed LS-EDA and speedup strategy based on latent space. Section 4 gives experimental results on standard test functions. Section 5 reveals the properties of the test functions by LS-EDA. We further verify the performance of the proposed LS-EDA on the problems from the CEC 2015 competition and then analyze its advantages and disadvantages. Section 6 gives conclusion and future directions.

2 Background

2.1 Cooperative coevolution (CC) strategies

Potter and De Jong (1994) first combine CC framework with GA (CCGA). Two strategies in CCGA are employed: one-dimensional based and splitting-in-half strategies. The former decomposes a D-dimensional problem into D one-dimensional subproblems, while the latter divides it into two $\frac{D}{2}$-dimensional subproblems. CCGA-1 and CCGA-2 (Potter and De Jong 1994) belong to the fixed grouping-based algorithms. They differ in that the former conducts the collaboration of subproblems by means of the best individuals of each subproblem solver, while the latter execute the collaboration by means of the randomly chosen individuals. Because grouping-dependent variables requires one to detect strong their interacting characteristics, an automated decomposition approach, differential grouping (DG), is introduced by Omidvar et al. (2014). DECC-G proposed by Yang et al. (2008a) and Yang et al. (2007) combines random grouping with DE-based CC to solve LSGOs with up to 1000 dimensions. The random grouping strategy tries to randomly split a high-dimensional problem into multiple low-dimensional subproblems. Each subproblem is optimized repeatedly by a DE with a certain number of objective function evaluations. However, the performance of DECC-G degenerates when the number of interdependent variables grows. Yang et al. (2008b) proposed a new multilevel CC algorithm (MLCC) to improve DECC-G (Yang et al. 2008a). MLCC employs a decomposer pool to store several decomposers of different group sizes, and each decomposer has a specific value to evaluate objective function. In order to improve the accuracy rate of the variable grouping, Omidvar et al. (2017) proposed a novel differential grouping algorithm (DECC-DG2). Yang et al. (2017) proposed a new CC framework (CCFR), and it can efficiently allocate computational resources among the subpopulations according to their dynamic contributions. Gomes et al. (2017) proposed a novelty-driven CC framework, and its novelty search technique can drive evolution toward behavioral novelty and diversity and avoid the counterproductive attraction to stable states in coevolution. Dong et al. (2013) proposed a novel EDA framework with model complexity control (EDA-MCC) to scale up continuous EDAs, which can solve LSGOs with up to 500 dimensions. Omidvar et al. (2011) proposed contribution-based cooperative coevolution (CBCC) that automatically allocates the available computational budgets to subproblem solvers according to their contribution to the improvement in the optimization result.

2.2 Gaussian-based EDA

Since EDA was proposed in the 1990s, its variants with different probability models have emerged. Because the proposed LS-EDA is also based on a Gaussian model (Dong and Zhou 2014), we only review the Gaussian-based EDAs. The performance of Gaussian-based EDAs heavily depends on how to determine the mean vector and covariance matrix (Lin et al. 2016). The primary difference among them is the number of parameters in their adopted Gaussian model and the number of Gaussian models. Besides a single Gaussian model, Gaussian mixture models are also adopted in some variants to solve multimodal optimization problems (Lu and Yao 2005). We focus on single Gaussian model EDAs, which fall into two classes (Pelikan et al. 2002):

2.2.1 Univariate models

Decision variables are assumed independent. Their joint probability is fully factorized (Zhou et al. 2015):

$$P(x_1, \ldots, x_D) = \prod_{i=1}^{D} \mathcal{N}(x_i), \tag{1}$$

where $\mathcal{N}(x_i)$ is the ith univariate Gaussian model with two parameters: mean and covariance. The examples of such methods are UMDA$_c^G$ (Larranaga and Lozano 2002) and PBIL$_c$ (Sebag and Ducoulombier 1998).

Univariate models can be found in the earliest EDAs. Let D denote the dimensionality of a search space. Total $2D$ parameters, i.e., D mean vectors and D covariances, need to be estimated for this kind of EDAs. Although the computation for estimating them is effectively reduced, their performance dramatically decreases when they are used to optimize complicated functions. Because they fail to accurately estimate the true distribution of selected individuals, especially in the cases of the dependent variables (Pelikan et al. 2002). This restricts their real-world applications.

2.2.2 Multivariate models

Multivariate interactions are proposed in Zhou et al. (2015):

$$P(x_1, \ldots, x_D) = \mathcal{N}(\mu, \Pi), \tag{2}$$

where $\mu \in \mathbb{R}^D$ and $\Pi \in \mathbb{R}^{D \times D}$ are a mean vector and covariance matrix, respectively. The representations of such methods are EMNA$_{\text{global}}$ (Larranaga and Lozano 2002), normal IDEA (Bosman and Thierens 2001; Bosman and Thierens 2000) and EGNA (Etxeberria and Lozano 2000).

Multivariate models can effectively capture the interdependencies among variables and provide useful information for guiding the global search on many unimodal and some multimodal problems. Because of this, many EDAs adopt this model, e.g., Hansen and Ostermeier (2001) and Bosman and Thierens (2001). In order to obtain μ and Π, a maximum likelihood estimate (MLE) is adopted in most of them. However, the MLE method requires large amounts of data to produce reliable estimates, which is affected by the well-known curse of dimensionality. That is, the amount of required data quadratically increases with the dimensionality of the search space, and as a result, the population size of EDAs quadratically grows with the problem dimensionality. Besides the high computation from MLE, subsequent sampling from Gaussian models needs plenty of computational resources. According to Hansen and Ostermeier (2001), it costs at least $O(D^3)$ to decompose a covariance matrix. Thus, the overall computational complexity of a multivariate Gaussian-based EDA is at least $O(D^3)$.

In summary, (1) a simple model, such as a univariate Gaussian model, cannot effectively capture the hidden statistical information in selected population because of its fewer degrees of freedom. It is easy to implement but insufficient to deal with complicated problems. (2) To build a good multivariate model, large population size is required and the learning process (including network structure learning and parameter learning) also is very time-consuming (Dong et al. 2013; Sun et al. 2003).

2.3 Probabilistic PCA and EM algorithm

For probabilistic model-based EDA, the selected samples have the property that the sample points lie close to a manifold with much lower dimensionality than the original data space. This naturally leads to such a generative models in which a point in a latent subspace is sampled and then transformed into an original space point, which is sampled from some conditional distribution given the latent variables. This process is well-known principal component analysis (PCA). PCA, known as the Karhunen–Loève transform, is a widely used technique for applications such as dimensionality reduction, feature extraction, data visualization and lossy data compression. PCA cannot be directly used to form a linear Gaussian latent variable model. Instead, a probabilistic PCA model can be used as one to generate samples for EDA.

A probabilistic PCA model is a Bayesian treatment of PCA and can be run generatively to provide samples from its distribution. Assume that both the latent and observed variables obey Gaussian distributions, we can obtain the simplest continuous latent variable model (Bishop 2006), and all of the conditional and marginal distributions are Gaussian in a probabilistic PCA. When latent variable vector z is explicitly introduced, the probabilistic PCA model can be formulated

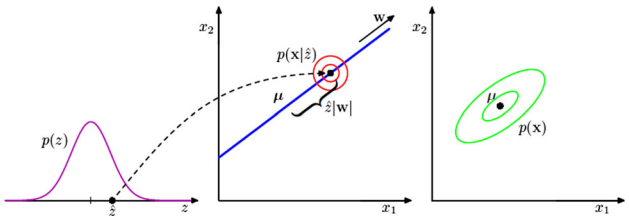

Fig. 1 A sample generated based on the probabilistic PCA model

as follows. Suppose that the prior distribution $p(z)$ over the latent principal component subspace obeys a Gaussian model with a zero mean and unit covariance. We can then have:

$$p(z) = \mathcal{N}(z|0, I_{M\times M}), \tag{3}$$

where z is the M-dimensional Gaussian latent variable vector, $I_{M\times M}$ is the $M \times M$ unit diagonal matrix. Suppose that the conditional distribution $p(x|z)$ for the observed variable x conditioned on the value of latent variables also obeys a Gaussian model, i.e.,

$$p(x|z) = \mathcal{N}(x|Wz + \mu, \sigma^2 I_{D\times D}), \tag{4}$$

where the $D \times M$ matrix W is the transformation matrix converting latent variables to observed variables, μ is the mean vector for observed variables, σ is the noise variance governing the variance of the conditional distribution and $I_{D\times D}$ is a $D \times D$ unit diagonal matrix. We can see that the mean of x is a linear function of M-dimensional vector z and D-dimensional vector μ.

The closed-form solutions for transformation matrix W is

$$W = U(L - \sigma^2 I), \tag{5}$$

where U is a $D \times M$ matrix whose columns are given by M eigenvectors of the data covariance matrix corresponding to the top M eigenvalues λ_i and L is an $M \times M$ diagonal matrix whose diagonal elements equal the corresponding eigenvalues λ_i's.

It is clear that a new sample for the observed variable in the probabilistic PCA model is generated as follows: The value of a latent variable is firstly sampled, and then, its corresponding observed variable is sampled conditioned on this latent value by the transformation of the M-dimensional latent variable vector z plus additive Gaussian "noise," i.e.,

$$x = Wz + \mu + \varepsilon, \tag{6}$$

where ε is a D-dimensional zero-mean Gaussian-distributed noise variable with covariance $\sigma^2 I$. This generation process is illustrated in Fig. 1, and the details can be found in Bishop (2006).

Algorithm 1 General Framework of an EM Algorithm

1: Initial θ^{old}
2: **repeat**
3: **E step** Evaluate $p(Z|X, \theta^{\text{old}})$
4: **M step** Evaluate $\theta^{\text{new}} = \arg\max_{\theta} Q(\theta, \theta^{\text{old}})$ Where $Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \, ln \, p(X, Z|\theta)$
5: $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$
6: **until** a stopping criterion is met

We can see that the linear Gaussian latent subspace model is fully determined by its parameters: W, μ and σ. Their exact closed-form solution (Bishop 2006) can be deduced, and the MLE and eigenvalue decomposition (EVD) can be used to estimate their values. Yet their computational complexity is at the order of D^3. Thus, an iterative expectation–maximization (EM) algorithm may have some computational advantages over such closed-form solution (Bishop 2006). The goal of the EM algorithm is to find maximum likelihood solutions for probability models with latent variables. We denote the set of all observed data by X, in which the nth row represents X_n^T, and similarly, we denote the set of all latent variables by Z, with a corresponding row Z_n^T. All the model parameters W, μ and σ are denoted by θ, and so the log-likelihood function is given by

$$\ln \; p(X|\theta) = \ln \left\{ \sum_Z p(X, Z|\theta) \right\}. \tag{7}$$

Our knowledge about the values of latent variables in Z is given via the posterior distribution $p(X, Z|\theta)$ only, and thus, we consider its expected value under the posterior distribution of latent variables. In the EM algorithm, we maximize this expectation in the M step. If the current estimation of the parameters is denoted θ^{old}, a pair of successive E and M steps produces a revised estimate θ^{new}. The algorithm is initialized by choosing some starting values θ_0 for the parameters. In the E step, we use the current parameter values θ^{old} to find the posterior distribution of latent variables given by $p(Z|X, \theta^{\text{old}})$. We then use this posterior distribution to find the expectation of the complete data log likelihood evaluated for some general parameter value θ. This expectation, denoted $Q(\theta, \theta^{\text{old}})$, is given by:

$$Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \; \ln \; p(X, Z|\theta). \tag{8}$$

In the M step, we determine the revised parameter estimate θ^{new} by maximizing this function:

$$\theta^{\text{new}} = \arg\max_{\theta} Q(\theta, \theta^{\text{old}}). \tag{9}$$

Algorithm 1 shows a general framework of an EM algorithm, and its details can be found in Guo et al. (2008).

3 Probabilistic PCA-based EDA

As discussed above, the univariate and multivariate Gaussian EDAs differ in performance and computational complexity due to difference in the number of parameters of their Gaussian models. The former involves lesser parameters so that it faces with lower computational budgets but fails to capture the dominant correlation among variables. Oppositely, the latter refers to more parameters, leading to higher computational cost but effectively capturing the dominant correlations. An open issue arises: Can we design a new method to merge their advantages but avoid their drawbacks? Note that probabilistic PCA (pPCA) is an ideal bridge between univariate and multivariate Gaussian models. It represents a constrained form of a Gaussian model in which the number of free parameters can be freely scaled up while allowing the model to capture the interdependency among variables. Thus, we propose latent subspace-based EDA framework (LS-EDA) to integrate the above-mentioned advantages, e.g., capturing the correlation and low computational cost. Its basic idea is as follows: We replace the traditional Gaussian model in EDA with a pPCA model and then introduce EM algorithm to estimate its parameters. During its iterative process, a new sample z is first drawn in the latent subspace and then changes it to a individual x in the original space conditioned on z.

3.1 LS-EDA procedure

The procedure of the proposed LS-EDA is shown in Algorithm 2. It has five procedures: initialization, sampling, selection and parameters estimation.

Lines 1–3 are the initialization step, where M is the number of latent variables, N is the size of population and ω is the mixture ratio between current population and preceding one. The mean variable of the pPCA model is μ, which is initialized as a zero vector. The transformation matrix from the latent subspace to the original space is W, which is initialized as the first M columns of an $D \times M$ unit matrix. The variance scale of the pPCA model σ is set to 1.

Lines 5–6 are the sampling step. In line 5, latent variables $Z_j = \{z_i, i = 1, 2, \ldots, M\}$, $j = 1, 2, \ldots, N$, are sampled from probabilistic distribution: $p(z_i) = \mathcal{N}(z_i|0, I)$. Subsequently, individuals $X_j = \{x_i, i = 1, 2, \ldots, D\}$, $j = 1, 2, \ldots, N$, in the original space are sampled from conditional probabilistic distribution: $p(x_i|z_i) = \mathcal{N}(x_i|Wz_i + \mu, (\sigma)^2 I)$, and each sample x_i conditionally depends on its corresponding latent variables z_i. τ is the scale factor on the domain of the benchmark functions.

Lines 7–11 are the selection steps. In the first generation, sample set D_l^{Se} which is used to estimate the pPCA model is set to the whole population; otherwise, it is composed of

Algorithm 2 Procedure of LS-EDA

1: Initial M, N, ω;
2: $\mu = zeros(D, 1), T = eye(D, D), W = T(:, 1 : M), \sigma = 1$;
3: Set $l = 0$;
4: **repeat**
5: Generate N samples $Z_j = \{z_i, i = 1, 2, \ldots, M\}, j = 1, 2, \ldots, N$ in the latent subspace complying with this probabilistic model: $p(z_i) = \mathcal{N}(z_i|0, I)$;
6: $D_l \leftarrow$ Generate N individuals $X_j = \{x_i, i = 1, 2, \ldots, D\}, j = 1, 2, \ldots, N$ in the original space complying with this probabilistic distribution: $p(x_i|z_i) = \mathcal{N}(x_i|\tau W z_i + \mu, (\tau\sigma)^2 I)$;
7: **if** $l > 0$ **then**
8: $D_l^{Se} \leftarrow$ Select best $\omega * N$ individuals from population D_{l-1} and best $(1 - \omega) * N$ individuals from population D_l;
9: **else**
10: $D_l^{Se} \leftarrow D_l$;
11: **end if**
12: $\mu \leftarrow$ calculate the mean value from the new population D_l^{Se} with the weight coefficients w in Eq. (10) ;
13: Learn parameters μ, W and σ of the probability distribution $p_l(x) = p(x|D_l^{Se})$ by the selected population D_l^{Se} by using the EM algorithm;
14: **if** $l\%100 = 0$ **then**
15: Calculate parameter M by Eq. (19);
16: **end if**
17: $l = l + 1$;
18: **until** a stopping criterion is met

the best $\omega * N$ individuals from population D_{l-1} and best $(1 - \omega) * N$ individuals from population D_l.

Lines 12–16 are the parameter estimation step. The first two lines show estimation for parameters μ, W, σ and M, and how to estimate them is given in Sects. 3.2 and 3.3.

3.2 Estimating parameters of pPCA

Sample set D_l^{Se} is replaced by combining the best individuals generated in the current population with the top ones from the previous generations, which can be used as an effective way to promote diversity in the population. In addition, the role of replacement methods in EDAs has shown that they can influence the accuracy of the probabilistic models learned by the algorithms.

The new mean vector μ is a weighted average of new selected population D_l^{Se}. The individuals in D_l^{Se} are sorted in descending order in terms of their function values, and then, the mean vector μ is calculated by

$$\mu_l = \sum_{i=1}^{N} w_i x_i^l, \quad \sum_{i=1}^{N} w_i = 1, w_1 \geq w_2 \geq \cdots \geq w_N \geq 0, \tag{10}$$

where N is the population size, $w \in \mathbb{R}^+$ (the nonnegative real number set), $i \in 1, 2, \ldots, N$ is the nonnegative weight coefficient for combination. Specially, we can set $w_{i=1,\ldots,N} = \frac{1}{N}$, which corresponds to the obtained result via MLE.

Because each data point x_i conditionally depends on a latent point z_i, the pPCA model can be expressed in terms of marginal distribution with respect to a continuous latent space z, and we can use MLE to find the model parameters. If we construct an exact closed-form solution by MLE, we should use eigenvalue decomposition to find the eigenvectors and eigenvalues of the data covariance matrix and then to evaluate W and σ. But the complexity of eigenvalue decomposition is approximately $O(D^3)$, leading high computational burden for high-dimensional problems. Therefore, we employ EM algorithm to find the model parameters as MLE. Such algorithm has many advantages, e.g., controllable computational budget and low chance for overflows, especially facing a high-dimensional space.

EM is an iterative approach. First, we compute the expectation of the log likelihood with respect to the posterior distribution by using "old" parameter values and then maximize the log likelihood to estimate the "new" parameter values. This process is repeated until the estimated parameters do not change (within given precision ε). The log-likelihood function based on a sample set takes the following form:

$$ln\ p(X, Z|\mu, W, \sigma^2) = \sum_{i=1}^{N} \{ln\ p(x_i|z_i) + ln\ p(z_i)\}, \tag{11}$$

where the ith row of the matrix Z is given by z_i and the mean vector μ is the sample mean $\bar{x}$. Instead of estimating u by MLE, we estimated it by Eq. (10). Considering the expectation of the posterior distribution over latent variables, we obtain:

$$\begin{aligned} E[ln\ p(X, Z|\mu, W, \sigma^2)] = -\sum_{i=1}^{N} \Bigg\{ & \frac{D}{2} ln(2\pi\sigma^2) \\ & + \frac{1}{2} Tr(E[z_i z_i^T]) + \frac{1}{2\sigma^2} \|x_i - \mu\|^2 \\ & - \frac{1}{\sigma^2} E[z_i]^T W^T (x_i - \mu) + \frac{1}{2\sigma^2} Tr(E[z_i z_i^T] W^T W) \Bigg\}. \end{aligned} \tag{12}$$

In the E step, the "old" parameters are used to calculate the expectation and variance of z_n, i.e.,

$$E[z_i] = M^{-1} W^T (x_i - \bar{x}), \tag{13}$$

$$E[z_i z_i^T] = \sigma^2 M^{-1} + E[z_i] E[z_i]^T. \tag{14}$$

In the M step, "new" parameter values are calculated as

$$W_{\text{new}} = \left[\sum_{i=1}^{N}(x_i - \bar{x})E[z_i]^T\right]\left[\sum_{i=1}^{N}E[z_i z_i^T]\right], \tag{15}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND}\sum_{i=1}^{N}\{\|x_i - \bar{x}\|^2 - 2E[z_i]^T W^T_{\text{new}}(x_i - \bar{x}) + Tr(E[z_i z_i^T]W^T_{\text{new}}W_{\text{new}})\}. \tag{16}$$

The EM algorithm iteratively executes E and M steps until convergence.

3.3 Determine the number of latent variables

The latent variable Z stands for the most important principal component of the current population D_l^{Se}, and it represents main search direction. In order to estimate its number M, the top-M eigenvalues should contain above 90% information. To do that, firstly, all the eigenvalues are estimated by Eqs. (17) and (18), and then, the top-M eigenvalues are determined by Eq. (19).

$$Au_i = \lambda_i u_i, \tag{17}$$

$$A = \sum_{i=1}^{D}\lambda_i u_i u_i^T, \tag{18}$$

where A is the $D \times D$ covariance matrix of D_l^{Se}, u_i is the eigenvector and λ_i is the corresponding eigenvalue, $i = 1, 2, \ldots, D$.

$$\frac{\sum_{i=1}^{M}\lambda_i}{\sum_{i=1}^{D}\lambda_i} \geq 90\%. \tag{19}$$

Because the high computational cost eigenvalue decomposition, M is estimated at every 100 generations.

3.4 Complexity analysis

Experimental results show that the runtime of LS-EDA is faster than multivariate Gaussian model-based EDAs models. The reason behind it is degrees of freedom and computational complexity.

First, we derive the relationship between a pPCA model and the one in the original space. Equations (3) and (4) show that the distribution of latent variables z and the conditional distribution of the original variable x are Gaussian models. Hence, this marginal distribution of x is also Gaussian, i.e.,

$$p(x) = \mathcal{N}(x|\mu, C), \tag{20}$$

where the $D \times D$ covariance matrix C is defined as

$$C = WW^T + \sigma^2 I. \tag{21}$$

While the pPCA model is based on a multivariate Gaussian distribution, its degrees of freedom and the number of independent parameters can be controlled by the dimensionality of a latent subspace. Meanwhile, the transformation matrix W from latent variables to original ones contains the most important directions in the data. Thus, the pPCA model can capture the latent dominant interdependency in the given data. A full-rank Gaussian model involves $D(D+1)/2$ independent parameters, D mean parameters and $D(D-1)/2$ covariance parameters. Thus, the number of degrees of freedom scales up quadratically with D, leading to excessive computational burden, especially for high dimensionality. The pPCA model reduces variance parameters by principal components in the transformation matrix W. Thus, it has D independent parameters in its mean μ, one parameter for σ, and $D \times M$ independent parameters in W, resulting in a total of only $(M+1) \times D + 1$ independent parameters. Consequently, the number of its parameters grows linearly with dimensionality given a fixed M.

Similar to the other iteration-based meta-heuristic algorithms, the computational time is evaluated by the runtime for each cycle multiplied by the number of iterations. From Algorithm 2, we can see that the sampling cost for each Gaussian-based EDA is equal, yet the time for estimating independent parameters varies. Thus, we focus on the computational time from EM algorithm. The most prominent benefit of EM algorithm for probability PCA is computational efficiency, especially for large-scale optimization problems. Each cycle of EM has low computation cost, and thus, EM-based pPCA is much more efficient than conventional PCA in case of high-dimensional problems. EM does not explicitly construct the original covariance matrix. Its most computationally demanding operation is the sums over the data set, which costs $O(NDM)$, where N is the population size. If the number of iterations of EM is k, the total computational cost is $O(kNDM)$. When k and N are set to be a fixed value and far less than D, and in general $M \ll D$, EM's actual computation time can be far less than $O(D^3)$ required by eigenvalue decomposition.

4 Experimental studies

4.1 Setup of experiments

In order to fully evaluate the performance of LS-EDA, we design four kinds of experiments, respectively, denoted by Experiments I, II, III and IV. Experiments I are aimed to test its parameters impact, also, i.e., sensitivity analysis. Experiments II are aimed to compare its scalability with the other state-of-the-art algorithms on the problems with differ-

Table 1 Test functions used in experiments

Func	Description	Expression	Domain
F1	Sphere (f1 in Yao et al. 1999)	$F(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
F2	Shifted Sphere (F1 in Suganthan et al. 2005)	$F(\vec{x}) = \sum_{i=1}^{n} z_i^2 + f_{bias1}, \vec{z} = \vec{x} - \vec{o}$	$[-100, 100]^n$
F3	Schwefel's Problem 2.21 (f4 in Yao et al. 1999)	$F(\vec{x}) = \max_i\{\|x_i\|, 1 \le i \le n\}$	$[-100, 100]^n$
F4	Shifted F3	$F(\vec{x}) = \max_i\{\|z_i\|, 1 \le i \le n\}, \vec{z} = \vec{x} - \vec{o}$	$[-100, 100]^n$
F5	Schwefel's Problem 2.21 (f4 in Yao et al. 1999)	$F(\vec{x}) = \sum_{i=1}^{n}[(x_1 - x_i^2)^2 + (x_i - 1)^2]$	$[-10, 10]^n$
F6	Shifted F5	$F(\vec{x}) = \sum_{i=1}^{n}[(z_1 - z_i^2)^2 + (z_i - 1)^2], \vec{z} = \vec{x} - \vec{o} + 1$	$[-10, 10]^n$
F7	Rosenbrock (f5 in Yao et al. 1999)	$F(\vec{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-100, 100]^n$
F8	Shifted Rosenbrock (F6 in Suganthan et al. 2005)	$F(\vec{x}) = \sum_{i=1}^{n-1}[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2] + f_{bias6}, \vec{z} = \vec{x} - \vec{o} + \vec{1}$	$[-100, 100]^n$
F9	Shifted Rotated High Conditioned Elliptic (F3 in Suganthan et al. 2005)	$F(\vec{x}) = \sum_{i=1}^{n}(10^6)^{\frac{i-1}{n-1}} z_i^2 + f_{bias3}, \vec{z} = (\vec{x} - \vec{o}) * M$	$[-100, 100]^n$
F10	Schwefel 2.6 with global optimum on bounds (F5 in Suganthan et al. 2005)	$F(\vec{x}) = \max\{\|A_i\vec{x} - B_i\|\} + f_{bias5}, i = 1, \ldots, n$	$[-100, 100]^n$
F11	Rastrigin (f9 in Yao et al. 1999)	$F(\vec{x}) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5, 5]^n$
F12	Shifted Rotated Rastrigin (F10 in Suganthan et al. 2005)	$F(\vec{x}) = \sum_{i=1}^{n}[z_i^2 - 10\cos(2\pi x_i) + 10] + f_{bias10}, \vec{z} = (\vec{x} - \vec{o}) * M$	$[-5, 5]^n$
F13	Shifted Expanded Griewank plus Rosenbrock (F13 in Suganthan et al. 2005)	$GF(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$ $RF(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$ $F(\vec{x}) = GF(RF(z_1, z_2)) + GF(RF(z_2, z_3)) + \cdots + GF(RF(z_{D-1}, z_D)) + GF(RF(z_D, z_1)) + f_{bias13}$ $\vec{z} = \vec{x} - \vec{o} + 1$	$[-3, 1]^n$

The domains of function F7 and F11 are changed from original definitions in Yao et al. (1999) to make the consistent with the domains of F8 and F12, respectively. F4 and F6 are shifted version of F3 and F5, respectively. The shifted global optima are generated following the same way of Suganthan et al. (2005)

ent dimensionality in terms of solution precision when the number of function evaluations is fixed for all algorithms. Experiments III are aimed to compare the performance of LS-EDA with state-of-the-art CC algorithms on the ones with high dimensionality. Experiments IV are aimed to compare the performance of LS-EDA on the CEC 2013 LSGO Competition.

4.1.1 Benchmark functions

Similar to Dong et al. (2013), we choose 13 benchmark functions from CEC2005 special session (Suganthan et al. 2005) and references (Etxeberria and Lozano 2000; Yao et al. 1999). All 13 functions are listed in Table 1. They fall into three groups: (1) F1 and F2 belong to separable unimodal problems; (2) F3–F10 belong to nonseparable problems with only a few (≤ 2) local optima; and (3) F11–F13 belong to multimodal problems. Some transformation strategies, such as symmetry breaking, irregularities and ill-conditional transformations, are introduced to increase their difficulties in Suganthan et al. (2005).

4.1.2 Parameter settings

We implement our method in MATLAB on an Intel(R) Core(TM) i3 3.60 GHz computer with 4 GB RAM. To be fair, for all algorithms, the terminated condition is set to same, that is, only a fixed number of maximum function evaluations (FEs $= 10^4 \times D$ in Suganthan et al. 2005). Each algorithm is independently performed for 25 times for the purpose of sta-

Table 2 The factors and levels of the LS-EDA

Levels	Factors	
	Mixed ratio ω	Population size N
1	0.3	30
2	0.5	50
3	0.7	100
4	0.9	200
5		500

tistical comparisons, and for each run, the difference between the estimated best fitness and the known global optimum, i.e., $F(\vec{x}) - F(\vec{x}^*)$, is recorded. Because all the functions are minimization problems, the smaller difference means the higher precision given an algorithm.

4.2 Experiments I: Sensitivity analysis

The key parameters of LS-EDA are mixed ratio ω and population size N. In the section of sensitivity analysis, we will discover how they impact the performance of LS-EDA.

4.2.1 Orthogonal experimental design

In LS-EDA, ω means how many best ranked individuals from the current population and the previous one to compose the selected population used for the estimation of parameters in its pPCA model and N means the population size. We use orthogonal experimental design to find the relationship among these parameters. The factors and levels of LS-EDA are given in Table 2, and the orthogonal array is given in Table 3. When $D \leq 300$, the maximum number of function evaluations is set to $10^4 \times D$. It is set to 2.5×10^6 for $D > 300$. Each experiment is performed for 25 independent runs. All experimental results are shown in Fig. 2.

Figure 2 shows that six combinations, i.e., 14, 9, 13, 8, 15 and 10, significantly outperform the others. In these six combinations, $\omega = 0.5$ and 0.7, $N = 100$, 200 and 500 lead to the best performance. The performance of LS-EDA degrades when ω is lower than 0.5 or larger than 0.9. When ω belongs to [0.5, 0.9], it is slightly sensitive to different ω.

Experimental results also show that the size of population in LS-EDA is set to 200; for all most cases, its performance can be obtained. Traditional EDAs (Mühlenbein and Paass 1996; Bielza et al. 2009; Wagner et al. 2004), often require a large size of population to obtain credible estimated parameters for their probabilistic model. However, when the maximum number of function evaluations is given, a large population size reduces the number of generations, thereby inadequately exploring the search space. On the contrary, although the number of generations may increase when we set a small size of population, it may lead to false or misleading estimation of the model parameters. As a result, we must make a good trade-off between population size and the number of generations and find how the former affects the performance of LS-EDA.

Table 3 The orthogonal array $L_{20}(4 \times 5)$

Combination	Factors	
	Mixed ratio ω	Population size N
1	0.3	30
2	0.3	50
3	0.3	100
4	0.3	200
5	0.3	500
6	0.5	30
7	0.5	50
8	0.5	100
9	0.5	200
10	0.5	500
11	0.7	30
12	0.7	50
13	0.7	100
14	0.7	200
15	0.7	500
16	0.9	30
17	0.9	50
18	0.9	100
19	0.9	200
20	0.9	500

4.3 Experiments II: Scalability of LS-EDA

4.3.1 Comparison with other state-of-the-art algorithms on 50-D and 100-D

In experiments II, six algorithms, UMDA$_c^G$ (Bielza et al. 2009), EMNA$_{\text{global}}$ (Bielza et al. 2009), EEDA (Wagner et al. 2004), EDA-MCC (Dong et al. 2013), CMA-ES (Hansen and Ostermeier 2001) and RP-EDA (Kabn et al. 2015) (Sanyang and Kabn 2015), are used for comparisons with LS-EDA. CMA-ES is a state-of-the-art evolutionary algorithm for continuous optimization. The other algorithms can be divided into three categories: univariate, multivariate and grouping-based Gaussian EDAs. UMDA$_c^G$ belongs to the first case, EMNA$_{\text{global}}$ and EEDA are the representative of the second case, and EDA-MCC and RP-EDA are in the last case, and MLE is used to estimate their parameters.

Because we fail to obtain the source codes of EMNA$_{\text{global}}$, EEDA and EDA-MCC, we adopt their results reported in Dong et al. (2013) for our comparison.

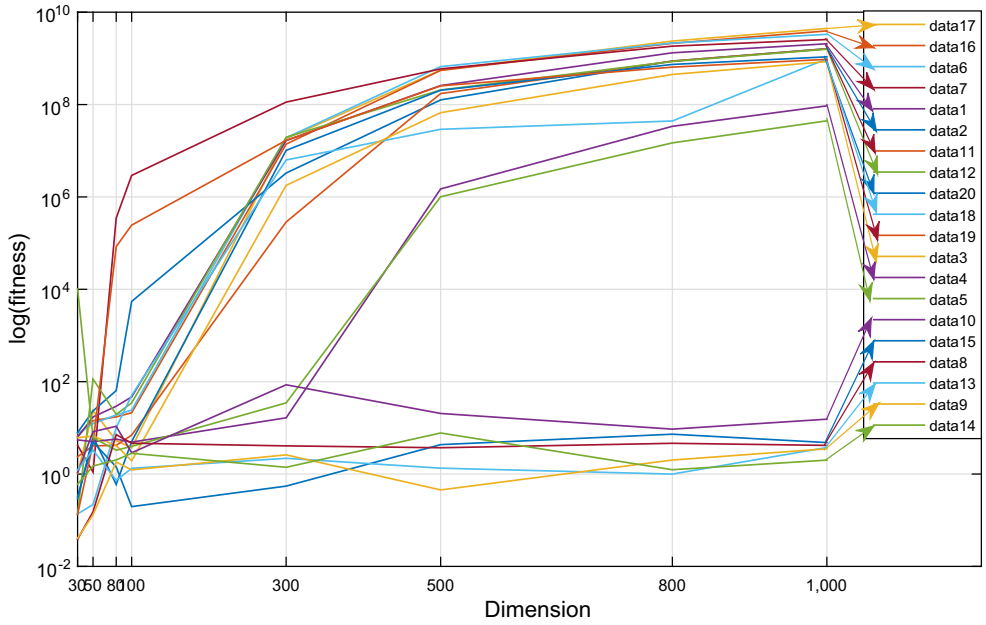

Fig. 2 Results of orthogonal design on 30D–1000D

The parameters of LS-EDA are given in Algorithm 2. For all test functions, we use a simple averaging the values of parameters. We take the mixed ratio $\omega = 0.7$ and the population size $N = 200$. The maximum number of function evaluations is same as the one in Suganthan et al. (2005), i.e., FEs $= 10^4 \times D$. The results are given in Tables 4– 5.

As in Dong et al. (2013), the difference between the estimated best fitness and the known global optimum, i.e., $F(\vec{x}) - F(\vec{x}^*)$, is recorded for comparisons. Since the difference is always nonnegative for minimization problems, the smaller it is, the better an algorithm performs. In Dong et al. (2013), if $F(\vec{x}) - F(\vec{x}^*)$ is smaller than $1e^{-12}$, it is regard as zero, i.e., the global optimum is reached.

(1) Separable unimodal problems Functions F1 and F2 are unimodal separable structures, and F2 is a shifting transformation of F1. Tables 4 and 5 show that UMDA$_c^G$, EDA-MCC, LS-EDA and CMA-ES can find the optima for them. However, EMNA$_{\text{global}}$, EEDA and RP-EDA show significant degradation in terms of their solution precision when the dimensionality increases. Since UMDA$_c^G$ is based on a univariable model and EDA-MCC is based on correlation grouping, their performance shows that they are suited for solving the univariable problems.

(2) Nonseparable problems with only a few local optima This group of functions either is unimodal or has a few local optima only, and F4, F6 and F8 are the shifted versions of F3, F5 and F7, respectively. Tables 4–5 show that LS-EDA shows overwhelming performance among all the compared algorithms, and it performs the best on all tests except F9 and F10. UMDA$_c^G$, EMNA$_{\text{global}}$ and EEDA have trouble to deal with this group of nonseparable functions, especially for the first two functions. The reason behind it may be the nonseparability of these test functions. It is in fact easy to understand why UMDA$_c^G$ performs badly, because it is univariable Gaussian-based EDA. EMNA$_{\text{global}}$ and EEDA are multivariable Gaussian-based EDAs. The reason for their performance degradation may be the insufficient or inaccurate estimations of their covariance matrix. EEDA performs better than EMNA$_{\text{global}}$ because the former can be seen as an extension to the latter. Note that, although EDA-MCC and CMA-ES perform worse than LS-EDA, their solution precisions are within the same order of magnitudes as the ones found by LS-EDA. In other words, we cannot conclude which one is statistically significantly superior to the other. Overall, among all compared algorithms, LS-EDA shows the most robust performance.

(3) Multimodal problems with many local optima Because functions F11–F13 are multimodal problems with abundant local optima and relatively complicated function landscape, the problems are hard to solve with most EDAs. F12 is the shifted version of F11, and F13 has many global and local optima. In this group of functions, the performance of UMDA$_c^G$ is superior to the other six algorithms on F11–F13. The performance of EEDA, EDA-MCC, CMA-ES and LS-EDA is similar to each other. CMA-ES performs better than EMNA$_{\text{global}}$, EEDA, EDA-MCC and LS-EDA on F13. Intuitively, it is hard to capture the shape by one Gaussian model due to multimodal landscape, and the experimental results coincide with this. EMNA$_{\text{global}}$, EEDA, EDA-MCC and LS-EDA belong to a single multivariate model. They have trouble to estimate their parameters for this group of functions. Since UMDA$_c^G$ employs multiple univariate Gaus-

Table 4 Comparisons of 50-D functions

Func	$UMDA_c^G$	$EMNA_{global}$	EEDA	EDA-MCC	CMA-ES	RP-EDA	LS-EDA
F1							
Mean	**0** ≈	1.30E−11	**0**	**0**	**0** ≈	**0** ≈	**0**
Std	**0**	6.30E−11	**0**	**0**	**0**	**0**	**0**
F2							
Mean	**0** ≈	4.50E+04	**0**	**0**	**0** ≈	1.57E−07 −	**0**
Std	**0**	2.20E+03	**0**	**0**	**0**	2.73E+04	**0**
F3							
Mean	2.60E−04 −	1.20E−01	1.80E−08	**0**	**0** ≈	1.73E−03 −	**0**
Std	1.50E−05	1.20E−01	2.40E−09	**0**	**0**	3.73E−04	**0**
F4							
Mean	3.40E+01 −	4.10E+01	1.40E−05	**0**	**0** ≈	3.77E−03 −	**0**
Std	2.50E+00	2.60E+00	6.80E−05	**0**	**0** ≈	1.59E+01	**0**
F5							
Mean	1.50E+01 −	1.50E+02	2.40E−02	**0**	**0** ≈	3.62E−02 −	**0**
Std	4.10E+00	1.40E+01	3.70E−03	**0**	**0**	**0**	**0**
F6							
Mean	1.40E+01 −	6.60E+03	1.00E−01	**0**	**0** ≈	5.68E+08 −	**0**
Std	5.20E+00	9.40E+02	1.20E−02	**0**	**0**	1.39E+06	**0**
F7							
Mean	4.80E+01 −	5.70E+01	5.00E+01	4.70E+01	**0** +	6.62E+01 −	**4.41E+00**
Std	3.40E−02	5.90E+00	9.20E+00	2.10E−01	**0**	3.94E+01	**9.93E−01**
F8							
Mean	4.10E+02 −	4.00E+09	5.20E+02	4.80E+01	**0** +	1.21E+03 −	7.79E+00
Std	9.10E+02	7.50E+08	1.00E+03	1.50E−01	**0**	1.68E+03	3.48E+00
F9							
Mean	4.30E+07 −	1.80E+09	4.10E+06	3.60E+06	**0** +	1.63E+07 −	2.08E+06
Std	4.10E+06	2.40E+08	1.40E+06	1.50E+06	**0**	3.17E+06	9.53E+05
F10							
Mean	4.90E+03 ≈	2.90E+04	2.00E+03	3.10E+03	**1.51E+03** ≈	4.31E+03 ≈	1.92E+03
Std	1.80E+02	1.40E+03	2.00E+02	3.40E+02	**4.89E+02**	3.77E+02	3.73E+02
F11							
Mean	**0** +	7.70E+00	3.10E+02	2.90E+02	6.69E+01 +	3.49E+02 ≈	3.12E+02
Std	**0**	5.00E+00	1.30E+01	1.40E+01	1.44E+01	1.81E+01	1.27E+01
F12							
Mean	**2.10E+00** +	3.20E+02	3.10E+02	3.00E+02	9.80E+01 +	3.44E+02 ≈	3.21E+02
Std	**9.50E−01**	2.10E+01	1.70E+01	1.46E+01	1.96E+01	8.46E+00	6.39E+00
F13							
Mean	**7.80E+00** +	9.90E+01	2.70E+01	2.60E+01	6.21E+00 +	3.04E+01 ≈	2.78E+01
Std	**8.30E−01**	2.40E+01	1.10E+00	9.20E−01	9.47E−01	1.54E+00	9.17E−01
F14							
Mean	3.93E+00−	–	–	–	**0**+	1.63E+00−	5.33E−01
Std	2.55E+00	–	–	–	**0**	1.89E+00	3.32E−01
−/+/≈	8/3/3	–	–	–	0/7/7	9/0/5	

For each problem, the best result is bolded. The result of $EMNA_{global}$, EEDA and EDA-MCC was reported in Dong et al. (2013). Results of LS-EDA are compared with those of $UMDA_c^G$, CMA-ES and RP-EDA, respectively, by Wilcoxon rank-sum test at the significance level of 0.05. The marker "−" is worse than the results of LS-EDA, "+" is better than the results of LS-EDA, and "≈" is equivalent to the results of LS-EDA

Table 5 Comparisons of 100-D functions

Func	$UMDA_c^G$	$EMNA_{global}$	EEDA	EDA-MCC	CMA-ES	RP-EDA	LS-EDA
F1							
Mean	**0** ≈	1.40E+01	**0**	**0**	**0** ≈	**0** ≈	**0**
Std	**0**	5.60E+00	**0**	**0**	**0**	**0**	**0**
F2							
Mean	**0** ≈	1.40E+05	5.30E−10	**0**	**0** ≈	4.96E−07 −	**0**
Std	**0**	4.00E+03	1.40E−09	**0**	**0**	3.31E−08	**0**
F3							
Mean	2.60E−02 −	3.30E+00	1.50E−03	**0**	**0** ≈	3.52E−04 −	**0**
Std	8.30E−02	7.00E−01	8.50E−04	**0**	**0**	7.11E−05	**0**
F4							
Mean	4.70E+01 −	5.80E+01	8.10E+00	**0**	**0** ≈	6.80E+00 −	**0**
Std	3.10E+00	2.70E+00	1.40E+00	**0**	**0**	1.89E+00	**0**
F5							
Mean	1.30E+02 −	6.70E+02	3.80E−01	**0**	**0** ≈	3.47E−02 −	**0**
Std	2.70E+01	7.50E+01	4.70E−02	**0**	**0**	6.15E−03	**0**
F6							
Mean	1.80E+02 −	2.20E+04	7.20E+00	**0**	**0** ≈	1.23E+09 −	**0**
Std	2.60E+01	2.10E+03	7.90E−01	**0**	**0**	2.45E+06	**0**
F7							
Mean	9.70E+01 −	2.70E+03	9.70E+01	9.60E+01	**0** +	1.42E+02 −	**1.33E+01**
Std	6.40E−02	1.50E+03	3.70E−01	7.50E−02	**0**	9.34E+01	**1.82E+01**
F8							
Mean	9.30E+02 −	1.80E+10	4.40E+04	9.60E+01	7.39E+01 −	9.78E+01 −	**8.80E+00**
Std	3.10E+03	1.90E+09	4.40E+04	1.30E−01	2.35E+00	8.16E−01	**9.93E−01**
F9							
Mean	4.30E+07 −	4.90E+08	2.20E+07	9.60E+06	**0** +	3.44E+07 −	4.75E+06
Std	3.10E+06	9.70E+07	3.70E+06	2.50E+06	**0**	9.70E+06	1.29E+06
F10							
Mean	5.90E+03 ≈	7.80E+04	4.40E+03	**1.90E+03**	2.97E+03 ≈	8.71E+03 ≈	2.91E+03
Std	4.30E+02	2.10E+03	6.00E+02	**3.60E+02**	4.07E+02	5.32E+02	5.54E+02
F11							
Mean	**0** +	1.40E+02	7.30E+02	7.50E+02	1.86E+02 +	7.90E+02 ≈	7.63E+02
Std	**0**	2.40E+01	1.50E+01	1.60E+01	5.01E+01	1.87E+01	2.26E+01
F12							
Mean	**8.60E+00** +	9.00E+02	7.30E+02	7.40E+02	2.68E+02 +	7.69E+02 ≈	7.65E+02
Std	**2.10E+00**	2.90E+01	2.50E+01	2.35E+01	2.44E+01	1.57E+01	1.94E+01
F13							
Mean	1.50E+01 +	1.20E+03	3.80E+01	6.50E+01	1.36E+01 +	6.92E+01 ≈	6.75E+01
Std	2.00E+00	1.90E+02	2.60E+01	1.60E+00	1.68E+00	1.35E+00	1.90E+00
F14							
Mean	3.93E+00−	–	–	–	**0**+	1.63E+00−	9.05E−01
Std	2.55E+00	–	–	–	**0**	1.89E+00	7.89E−01
−/+/≈	8/3/3	–	–	–	1/6/7	9/0/5	

For each problem, the best result is bolded. The result of $EMNA_{global}$, EEDA and EDA-MCC was reported in Dong et al. (2013). Results of LS-EDA are compared with those of $UMDA_c^G$, CMA-ES and RP-EDA, respectively, by Wilcoxon rank-sum test at the significance level of 0.05. The marker "−" is worse than the results of LS-EDA, "+" is better than the results of LS-EDA, and "≈" is equivalent to the results of LS-EDA

sian models, it thus performs well in this case. A previous study (Dong and Yao 2008) has shown that it cannot significantly improve the performance of these multivariate Gaussian-based EDAs. The huge number of local optima can easily mislead multivariate search and covariance matrix scaling. Perhaps, EDAs based on Gaussian mixture models can perform well for this group of functions. To summarize, as the no-free-lunch (NFL) theorem states, no algorithm is a panacea. Although the multivariate Gaussian EDAs, such as $\text{EMNA}_{\text{global}}$, EEDA, EDA-MCC and LS-EDA, outperform the univariate Gaussian EDAs, such as UMDA_c^G on F1–F10, they are indeed worse than the latter. A multivariate Gaussian EDA does not necessarily outperform a univariate one on F11–F13.

4.3.2 Comparison with other state-of-the-art algorithms on 500-D and 1000-D

As mentioned above, LSGOs have received more and more attentions in the intelligent optimization field. Thus, in this section we try to study the performance of LS-EDA on large-scale problems. The benchmark functions are still F1–F13, but their dimensionality is further enlarged to 500-D and 1000-D. Due to the unacceptable calculations of $\text{EMNA}_{\text{global}}$ and EEDA on any functions whose dimensionality is over 500, we choose UMDA_c^G, SaNSDE, EDA-MCC, CMA-ES, sep-CMA-ES and RP-EDA as its peers. Because the space complexity and internal time complexity of CMA-ES are quadratic, we only run it on 500-D. Since UMDA_c^G is univariate Gaussian EDAs, its computing budget linearly increases with the dimensionality. SaNSDE is a hybrid differential evolution (DE) algorithm, which combines the neighborhood search of NSDE and the self-adaptive mechanism of SaDE in a single algorithm. EDA-MCC and RP-EDA group all the variables into several subsets despite their different grouping strategies. CMA-ES performs the self-adaptation of the full covariance matrix as its evolution strategy. As variants of CMA-ES, RP-EDA has an ensemble of random projections to low dimensions of the set of the fittest search points as a basis for developing a new and generic divide-and-conquer methodology.

Because we have no source codes of EDA-MCC, we use its results reported in Dong et al. (2013) for 500-D for comparison. To play fair, as in Yang et al. (2008a), we set the maximum number of FEs is set to 2.5×10^6 for 500-D and 1000-D problems. The key parameters of the other algorithms are as follows: For UMDA_c^G, population size $N = 2000$ and selected best population size $m = 1000$. The population size of CMA-ES (Hansen and Ostermeier 2001) is set $\lambda = 4 + \lfloor 3 * \ln(D) \rfloor$. For RP-EDA (Kabn et al. 2015), $N = 300$, the random subspace dimension $k = 3$ and the number of subspaces is set to $M = 4 * \lceil D/k \rceil$. For the implemented details and parameter settings of other algorithms, please refer to Larranaga and Lozano (2002), Hansen and Ostermeier (2001) and Kabn et al. (2015). In LS-EDA, we set $N = 200$, $\omega = 0.7$. The experimental results are summarized in Table 6 for 500-D and Table 7 for 1000-D. For all compared algorithms except for EDA-MCC, the average evolutionary curve from 25 independent runs is plotted in Figs. 4 and 5 for 500-D and 1000-D, respectively. The average CPU time of 25 runs for 500-D problems is shown in Fig. 3. First, we compare convergence speeds for all algorithms. The same phenomena are shown in Figs. 4 and 5. The convergence speed of LS-EDA is the fastest one among all the compared algorithms except CMA-ES. For F1, the progress rate of LS-EDA is far more quickly than the compared algorithms, and it seems that the others are trapped into stagnation. For F4, F6 and F8, each of LS-EDA's convergence curves for these functions can be approximated by a hyperbolic curve, while the others seem to fall into trouble with more FEs. For F9 and F10, its evolutionary curves show the fast approaching the best solution at the very beginning, while the curves of the other algorithms show much slower convergence than those of LS-EDA.

Next, we compare the CPU time for all algorithms. Figure 3 clearly shows that CMA-ES is the most time-consuming algorithm, followed by UMDA_c^G and RP-EDA. Their CPU time is several orders of magnitude higher than its peers'. We can find that LS-EDA costs the least CPU time for most of the benchmark functions except for F6–F8, and its average CPU time is similar to that of SaNSDE in F7. Overall, sep-CMA-ES and SaNSDE cost CPU time more than LS-EDA. As a conclusion, in this round, LS-EDA, sep-CMA-ES and SaNSDE are winners, while UMDA_c^G, CMA-ES and RP-EDA are the losers. An interesting question is: Despite that UMDA_c^G, RP-EDA and LS-EDA are Gaussian-based EDA, why are UMDA_c^G and RP-EDA the loser? The answer may be the power of the EM algorithm employed in LS-EDA. We also verify that the advantage of LS-EDA is more significant than that of the others when D increases. Due to space limitation, we do not give the experimental results.

Finally, the solution precision of all compared algorithms for 500-D and 1000-D problems is analyzed. The results for 500-D problems are given in Table 6. It can be seen that UMDA_c^G, EDA-MCC, CMA-ES, sep-CMA-ES and LS-EDA perform perfectly on the first group of separable functions F1 and F2. They approach the global optima regardless of the average or best case, and SaNSDE and RP-EDA are slightly poorer than them. On the second group of nonseparable functions F3–F10, LS-EDA, EDA-MCC, CMA-ES and sep-CMA-ES show the most stable performance. They break even on F5 and F6, and both reach the global optimum. On the other functions, they have their merits as well. Although CMA-ES slightly outperforms LS-EDA on three functions F3, F4, and F9, it consumes much more time and memory. On the other functions, their precision is within

Table 6 The results of 500-D tests

Func	UMDA$_c^G$	SaNSDE	EDA-MCC	CMA-ES	sep-CMA-ES	RP-EDA	LS-EDA
F1							
Mean	0 ≈	1.33E−11 −	**0**	**0** ≈	**0** ≈	**0** ≈	**0**
Std	**0**	5.45E−12	**0**	**0**	**0**	**0**	**0**
F2							
Mean	0 ≈	1.10E−11 −	**0**	**0** ≈	**0** ≈	5.19E−06 −	**0**
Std	**0**	6.12E−12	**0**	**0**	**0**	3.92E−07	**0**
F3							
Mean	9.88E+01 −	4.05E+01 −	2.79E−01	**2.02E−06 +**	9.93E+01 −	3.16E+00 ≈	2.22E+00
Std	2.90E+00	4.21E+00	2.30E−02	**5.72E−07**	3.14E+01	8.43E−01	7.92E−01
F4							
Mean	6.46E+01 −	1.03E+02 −	3.27E−01	**1.42E−04 +**	1.62E+02 −	2.73E+01 −	9.84E+00
Std	4.20E+00	9.48E+01	3.70E−02	**4.24E−05**	7.63E+01	1.64E+00	1.62E+00
F5							
Mean	1.08E+03 −	4.26E−07 −	**0**	**0** ≈	**0** ≈	2.46E−02 −	**0**
Std	2.80E+02	4.16E−08	**0**	**0**	**0**	3.55E−03	**0**
F6							
Mean	7.33E+03 −	1.15E−06 −	**0**	**0** ≈	**0** ≈	6.34E+09 −	**0**
Std	8.70E+02	8.32E−07	**0**	**0**	**0**	1.21E+07	**0**
F7							
Mean	4.93E+02 ≈	1.32E+03 −	6.42E+02	3.45E+02 ≈	3.43E+02 ≈	5.17E+02 ≈	**3.35E+02**
Std	1.40E+01	9.64E+02	4.10E+02	8.26E+01	9.62E+01	5.52E+01	**7.02E+01**
F8							
Mean	9.71E+02 −	1.25E+03 −	6.77E+02	3.53E+02 ≈	3.11E+02 ≈	2.17E+03 −	**3.23E+02**
Std	9.80E+01	1.01E+02	6.30E+02	4.57E+01	9.17E+01	3.30E+03	**4.85E+01**
F9							
Mean	3.12E+08 −	5.90E+08 −	8.03E+07	**5.21E+03 +**	4.09E+07 ≈	2.42E+08 −	1.37E+07
Std	3.12E+07	3.74E+07	1.10E+07	**6.94E+02**	1.10E+07	3.97E+07	3.09E+06
F10							
Mean	4.28E+04 ≈	5.21E+05 −	2.09E+04	**1.00E+04** ≈	7.28E+04 ≈	4.96E+04 ≈	2.75E+04
Std	8.40E+02	4.03E+04	1.30E+03	**8.65E+02**	2.39E+04	1.70E+03	2.27E+03
F11							
Mean	**2.45E+00 +**	3.04E+02 +	5.24E+03	1.70E+03 ≈	2.63E+03 ≈	2.89E+02 +	4.80E+03
Std	**1.20E−01**	9.41E+01	3.90E+01	1.23E+02	8.13E+02	1.72E+01	4.04E+01
F12							
Mean	**7.68E+01 +**	6.97E+03 ≈	5.25E+03	2.24E+03 ≈	2.44E+03 ≈	3.10E+02 +	4.85E+03
Std	**6.50E+00**	2.68E+01	4.20E+01	8.47E+01	7.38E+02	2.10E+01	5.17E+01
F13							
Mean	8.12E+01 +	2.53E+02 ≈	4.52E+02	9.45E+01 +	1.06E+02 ≈	**5.74E+01 +**	4.10E+02
Std	3.10E+00	8.36E+00	5.00E+00	5.97E+00	6.20E+01	**2.56E+00**	5.90E+00
−/+/≈	7/3/4	11/1/2	−/−/−	0/5/9	2/0/11	7/3/4	

For each problem, the best result is bolded. The result of EDA-MCC was reported in Dong et al. (2013). Results of LS-EDA are compared with those of UMDA$_c^G$, SaNSDE, CMA-ES, sep-CMA-ES and RP-EDA, respectively, by Wilcoxon rank-sum test at the significance level of 0.05. The marker "-" is worse than the results of LS-EDA, "+" is better than the results of LS-EDA, and "≈" is equivalent to the results of LS-EDA

the same order of magnitude. RP-EDA are rather sensitive to the shifted global optimum. On the shifted functions F4, F6, and F8, LS-EDA performs well by holding almost the same performance, whereas RP-EDA becomes much worse. LS-EDA is not sensitive to the shifted and rotated function landscape, while RP-EDA is. Clearly, UMDA$_c^G$ performs very well on the last group of functions (F11–F13) with many local optima. It is among the best one of all the com-

Table 7 Comparisons of 1000-D tests

Func	$UMDA_c^G$	SaNSDE	EDA-MCC	sep-CMA-ES	RP-EDA	LS-EDA
F1						
Mean	1.76E−09 −	6.97E+00 −	3.54E−12 −	4.57E−84 ≈	2.59E−38 −	**3.40E−84**
Std	8.21E−10	3.12E−01	8.61E−13	8.24E−85	3.89E−39	**1.12E−84**
F2						
Mean	4.67E−09 −	1.60E+02 −	4.12E−10 −	1.84E−21 −	1.23E−05 −	**2.62E−23**
Std	1.02E−10	7.21E+01	9.67E−11	7.36E−22	7.38E−07	**8.24E−24**
F3						
Mean	9.91E+01 −	4.99E+01 ≈	3.56E+01 ≈	1.00E+02 −	2.72E+01 ≈	**2.67E+01**
Std	6.34E+01	1.04E+00	9.91E+00	5.12E+01	5.83E+00	**1.02E+01**
F4						
Mean	1.83E+02 −	1.17E+02 −	1.02E+02 −	1.67E+02 −	**3.82E+01** ≈	6.45E+01
Std	8.27E+01	7.23E+01	8.26E+01	8.32E+01	**1.19E+00**	3.68E+01
F5						
Mean	3.32E+03 −	1.09E+01 −	6.82E−12 −	6.67E−24 −	5.42E−01 −	**3.92E−26**
Std	1.06E+03	8.12E+00	2.73E−12	2.59E−24	3.15E−02	**2.03E−26**
F6						
Mean	1.84E+04 −	3.16E+01 −	7.63E−04 −	1.77E−20 −	1.28E+10 −	**4.01E−26**
Std	9.21E+03	9.78E+00	2.44E−04	6.31E−21	2.11E+07	**1.89E−26**
F7						
Mean	1.04E+03 ≈	3.31E+04 −	7.28E+03 −	9.33E+02 ≈	1.35E+03 ≈	**1.00E+03**
Std	8.23E+02	8.56E+03	3.14E+02	3.85E+02	8.33E+02	**7.23E+02**
F8						
Mean	1.45E+03 ≈	3.96E+06 −	7.14E+03 −	1.13E+03 ≈	1.56E+03 ≈	**1.03E+03**
Std	7.98E+02	1.03E+06	1.06E+02	6.17E+02	7.67E+02	**8.12E+02**
F9						
Mean	1.29E+09 −	5.79E+08 −	4.81E+08 −	8.88E+07 ≈	6.83E+08 −	**6.69E+07**
Std	8.41E+08	1.11E+07	2.19E+08	2.40E+07	5.48E+07	**3.24E+07**
F10						
Mean	1.45E+05 ≈	1.99E+05 ≈	4.36E+06 −	1.43E+05 ≈	**9.15E+04 +**	1.32E+05
Std	8.36E+04	8.42E+04	1.04E+06	7.19E+04	**3.98E+03**	9.78E+04
F11						
Mean	9.83E+03 +	8.69E+02 +	3.87E+04 −	5.79E+03 +	**7.17E+02 +**	1.09E+04
Std	6.71E+00	4.23E+02	9.27E+03	1.47E+03	**6.05E+01**	8.01E+03
F12						
Mean	1.14E+04 +	8.84E+03 +	4.57E+05 −	5.51E+03 +	**8.78E+02 +**	1.28E+04
Std	8.69E+00	4.10E+03	1.11E+05	1.03E+03	**6.82E+01**	9.42E+03
F13						
Mean	2.58E+05 −	6.27E+02 ≈	1.08E+03 ≈	2.60E+02 ≈	**1.16E+02** ≈	9.62E+02
Std	9.73E+01	3.57E+02	5.49E+02	8.31E+01	**3.01E+00**	6.13E+02
−/+/≈	8/2/3	8/2/3	11/0/2	5/2/6	5/3/5	

For each problem, the best result is bolded. Results of LS-EDA are compared with those of $UMDA_c^G$, SaNSDE, EDA-MCC, sep-CMA-ES and RP-EDA, respectively, by Wilcoxon rank-sum test at the significance level of 0.05. The marker "−" is worse than the results of LS-EDA, "+" is better than the results of LS-EDA, and "≈" is equivalent to the results of LS-EDA

pared algorithms. On F12 and F13, the results are consistent with previous observations. In general, LS-EDA performs statistically better than SaNSDE, $UMDA_c^G$, EDA-MCC and RP-EDA on 500-D problems, and the performances of CMA-ES, sep-CMA-ES and LS-EDA are similar with each other.

In order to test the scalability of LS-EDA for large-scale problems, we further enlarge the problem size of F1–F13 to

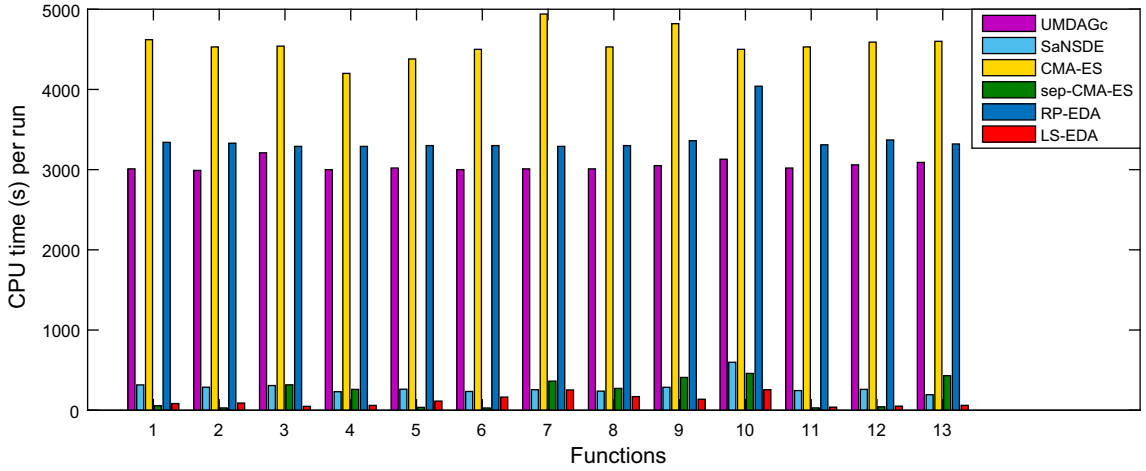

Fig. 3 Comparison of average CPU time on 500D

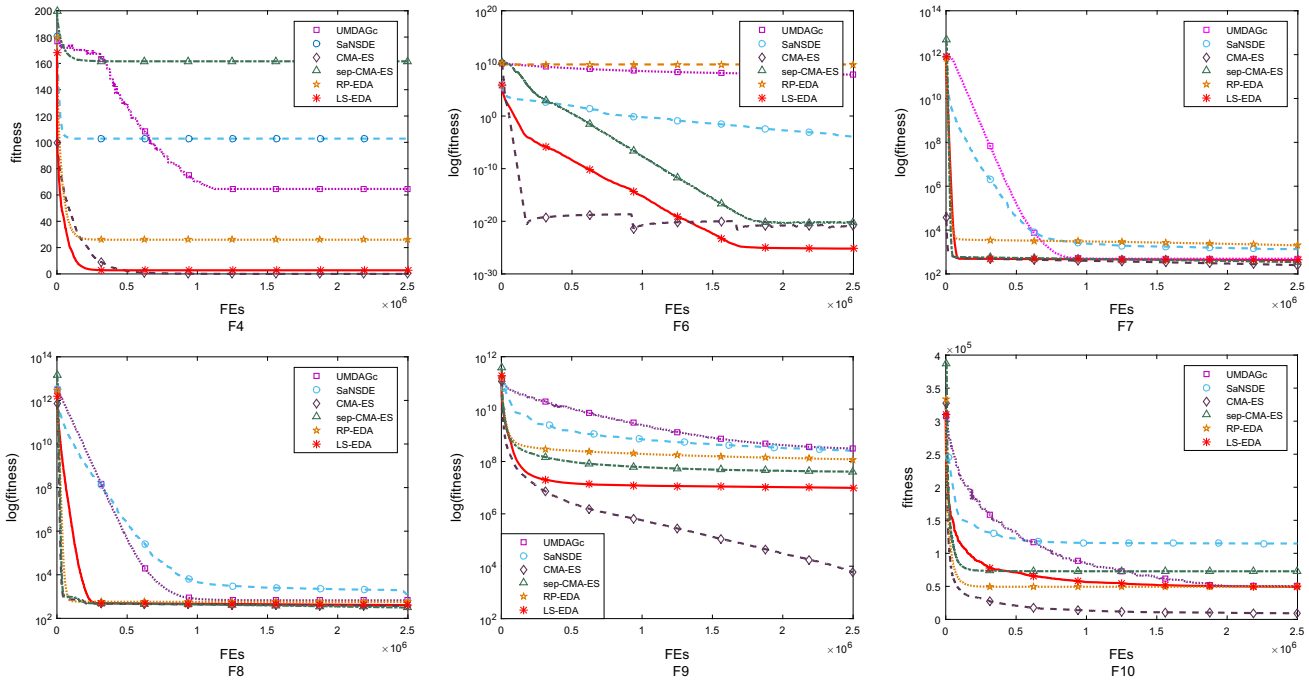

Fig. 4 Evolutionary curves on 500-D F4 and F6–F10

1000-D. Five algorithms are involved in experimental comparisons: UMDA$_c^G$, SaNSDE, EDA-MCC, sep-CMA-ES and RP-EDA. Because the results of EDA-MCC on 1000-D problems are not given in Dong et al. (2013), we have written the code of EDA-MCC base on Dong et al. (2013) by ourselves.

In Table 7, for 1000-D problems, it is clear that LS-EDA ranks the best since it outperforms the remaining ones on eight out of total 14 benchmark functions. RP-EDA ranks the second, since it outperforms LS-EDA on F10, F12 and F13, whereas LS-EDA significantly outperforms RP-EDA on five functions F1, F2, F5, F6 and F9. Generally speaking, LS-EDA outperforms UMDA$_c^G$, SaNSDE, EDA-MCC on eight functions F1, F2, F4–F6, F8–F10. Although sep-CMA-ES performs better than LS-EDA on F10 and F11, it becomes much worse on F2–F6. It is rather sensitive to the shifted and rotated function landscape. Compared with 500-D problems, LS-EDA shows more potential on 1000-D problems.

As a conclusion, LS-EDA shows robust performance on these 500-D and 1000-D problems, especially on nonseparable problems with only a few local optima. It performs statistically better than SaNSDE, UMDA$_c^G$, EDA-MCC, sep-CMA-ES and RP-EDA. CMA-ES performs generally well in low-dimensional problems, but it consumes much more time when the problem dimensionality becomes larger. Moreover,

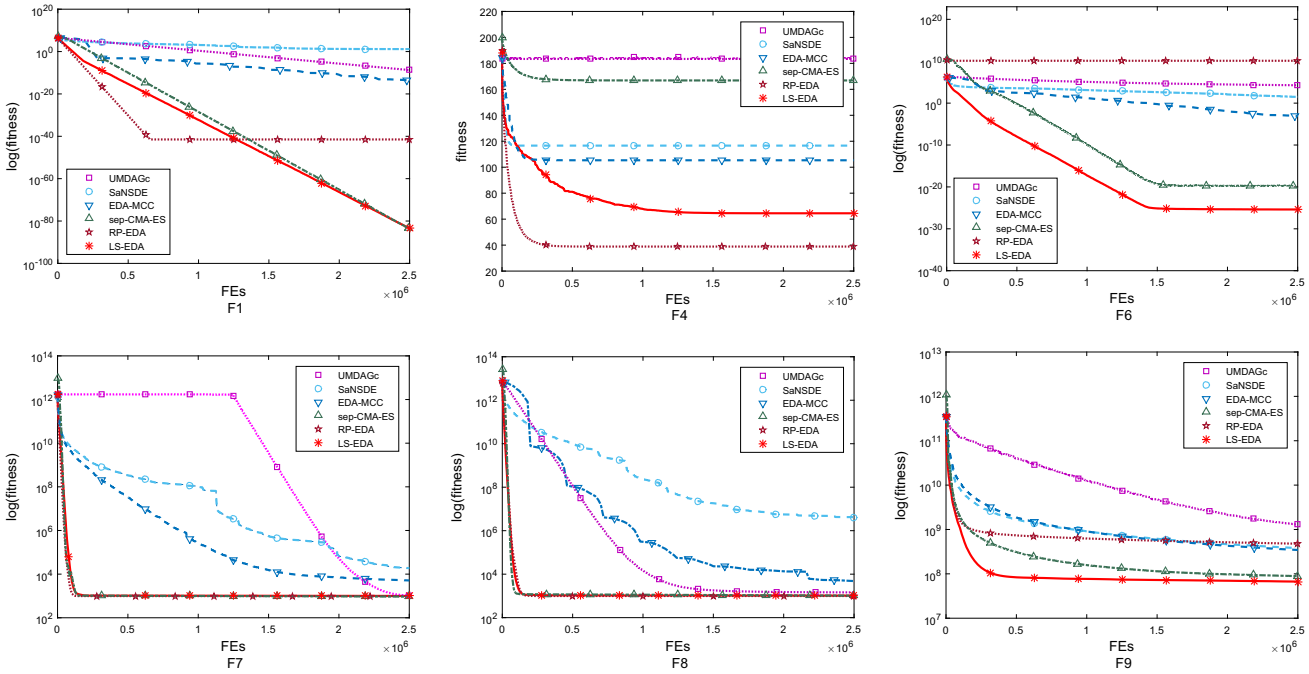

Fig. 5 Evolutionary curves on 1000-D F1, F4 and F6–F9

compared with other evolutionary algorithms, EDA-MCC, UMDA$_c^G$ and RP-EDA show their significant superiority on eight out of 13 functions, implying the advantages of using probabilistic models and statistical learning for optimization. Also note that our work has not tuned the parameters of LS-EDA further on specific problems. Evidently, LS-EDA performs well on the 1st and 2rd groups, but fails to do so on the 3rd groups. Although the accuracy of LS-EDA is lower than the other algorithms on the third group of three functions, experimental results show that its potential performance can be even better when the problem dimensionality becomes larger, which sharply differs from its peers.

4.4 Experiments III: Comparison with other CC algorithms on 500-D and 1000-D

Cooperative coevolution (CC) strategies are the main approach to solve LSGOs. In this section, we choose two state-of-the-art CC algorithms for comparisons: DECC-G (Yang et al. 2008a) and DECC-DG (Omidvar et al. 2014). The key parameters of the two algorithms are as follows: The population size of DECC-G, DECC-DG and SaNSDE is set to 100, its subcomponent dimension is 100 for all tests, and the number of cycles in DECC-G is 50. The implemented details and parameter settings of other algorithms can be found in Yang et al. (2008a) and Omidvar et al. (2014).

Tables 8 and 9 show that LS-EDA ranks the best among the compared algorithms. LS-EDA outperforms DECC-G and DECC-DG on eight functions on 500D and nine functions on 1000D. These two algorithms only outperform LS-EDA on F3 and F11 and similar to LS-EDA on the first group of benchmark functions, F1 and F2. DECC-G and DECC-DG embed SaNSDE into a cooperative coevolution framework to solve LSGOs through problem decomposition. But, LSGOs may have many complex interdependence between decision variables, and some problems are partially separable functions that consist of multiple independent subcomponents or fully nonseparable functions. So, in the CC strategy, the problem decomposition may greatly affect the performance of optimization algorithms in continuous domains.

4.5 Experiments IV: Comparisons on CEC 2013 LSGO competition

In order to test the performance of LS-EDA more fully, we further compare LS-EDA with DECC-G on the CEC2013 LSGO benchmark suite. Details of the benchmark suite of scalable functions and requirements on the simulation procedure are available in a technical report (Tang et al. 2013). There are total 15 continuous optimization functions with 1000 dimensions and different degrees of separability, from completely separable functions to fully nonseparable ones. They fall into four categories: (1) fully separable functions f1–f3; (2) partially separable functions f4–f7 and without f8–f11 separable subcomponents; (3) overlapping functions f12–f14; (4) nonseparable function f15.

Table 8 Comparisons with CC algorithms on 500-D

Func	DECC-G	DECC-DG	LS-EDA
F1			
Mean	**0** ≈	**0** ≈	**0**
Std	**0**	**0**	**0**
F2			
Mean	**0** ≈	**0** ≈	**0**
Std	**0**	**0**	**0**
F3			
Mean	**2.91E–05** +	4.33E+01 −	2.22E+00
Std	**3.21E–06**	9.86E+00	7.92E–01
F4			
Mean	8.08E+01 −	9.86E+01 −	**9.84E+00**
Std	4.21E+00	2.41E+01	**1.62E+00**
F5			
Mean	7.82E–08 −	4.07E–06 −	**0**
Std	2.34E–09	8.24E–07	**0**
F6			
Mean	3.88E–08 −	2.07E–03 −	**0**
Std	6.32E–09	7.39E–04	**0**
F7			
Mean	4.89E+02 ≈	1.38E+03 −	**3.35E+02**
Std	1.36E+01	6.26E+02	**7.02E+01**
F8			
Mean	1.81E+03 −	1.80E+09 −	**3.23E+02**
Std	8.24E+02	5.18E+08	**4.85E+01**
F9			
Mean	9.76E+07 −	9.25E+07 −	**1.37E+07**
Std	1.04E+06	3.19E+07	**3.09E+06**
F10			
Mean	1.23E+05 −	1.14E+05 −	**2.75E+04**
Std	8.63E+04	5.28E+04	**2.27E+03**
F11			
Mean	**0** +	3.77E+03 ≈	4.80E+03
Std	**0**	9.19E+02	4.04E+01
F12			
Mean	5.28E+03 ≈	8.93E+03 ≈	**4.85E+03**
Std	1.87E+01	2.60E+03	**5.17E+01**
F13			
Mean	1.71E+02 ≈	**1.19E+01** +	4.10E+02
Std	9.12E+00	**7.25E+00**	5.90E+00
−/+/≈	6/2/5	8/1/4	

Table 9 Comparisons with CC algorithms on 1000-D

Func	DECC-G	DECC-DG	LS-EDA
F1			
Mean	2.17E–25 −	2.55E–05 −	**3.40E–84**
Std	9.24E–26	8.16E–06	**1.12E–84**
F2			
Mean	1.55E–08 −	7.17E–04 −	**2.62E–23**
Std	8.35E–09	1.23E–04	**8.24E–24**
F3			
Mean	**1.01E–01** +	4.87E+01 ≈	2.67E+01
Std	**7.84E–02**	9.62E+00	1.02E+01
F4			
Mean	8.65E+01 ≈	1.05E+02 −	**6.45E+01**
Std	5.61E+01	6.14E+01	**3.68E+01**
F5			
Mean	1.19E–03 −	3.98E+03 −	**3.92E–26**
Std	9.32E–04	8.23E+02	**2.03E–26**
F6			
Mean	8.48E–04 −	4.73E+03 −	**4.01E–26**
Std	5.12E–04	8.91E+02	**1.89E–26**
F7			
Mean	**9.87E+02** ≈	5.13E+03 ≈	1.00E+03
Std	**4.25E+02**	1.21E+03	7.23E+02
F8			
Mean	4.78E+03 ≈	1.20E+11 −	**1.03E+03**
Std	1.34E+02	6.34E+10	**8.12E+02**
F9			
Mean	1.37E+09 −	3.45E+08 −	**6.69E+07**
Std	8.79E+08	8.14E+07	**3.24E+07**
F10			
Mean	2.38E+05 ≈	2.56E+05 ≈	**1.32E+05**
Std	9.87E+04	7.38E+04	**9.78E+04**
F11			
Mean	**3.55E–16** +	2.32E+04 ≈	1.09E+04
Std	**1.08E–16**	6.31E+03	8.01E+03
F12			
Mean	1.47E+04 ≈	2.22E+04 ≈	**1.28E+04**
Std	8.71E+03	7.50E+03	**9.42E+03**
F13			
Mean	4.65E+02 ≈	**4.61E+02** ≈	9.62E+02
Std	1.67E+02	**9.17E+01**	6.13E+02
−/+/≈	5/2/6	7/0/6	

To satisfy the specific requirements of contest, we set common parameters of LS-EDA as follows: (1) Stop criterion: It stops when the maximum number of FEs is reached. For every 1000-D function, FEs = 3.0e+06. (2) Statistics conditions: Each algorithm runs 25 times for each function, and the best, worst, average, medium results and standard deviation of 25 samples are used as comparison criteria. (3) The benchmarking algorithm: DECC-G. The population size of DECC-G is 100 and its subcomponent dimension is 100 for all tests. Since DECC-G employs SaNSDE as an optimizer of a collaborative framework, the parameters of SaNSDE can be found in Yang et al. (2008a). In LS-EDA, we set $M = 3$,

Table 10 Comparative results with DECC-G on CEC 2013 LSGO benchmark suite

Algorithms	f1	f2	f3	f4	f5	f6	f7	f8
DECC-G								
Best	**1.75E–13**	**9.90E+02**	**2.63E–10**	**7.58E+09**	7.28E+14	**6.96E–08**	**1.96E+08**	**1.43E+14**
Median	**2.00E–13**	**1.03E+03**	**2.85E–10**	**2.12E+10**	7.28E+14	**6.08E+04**	4.27E+08	**3.88E+14**
Worst	**2.45E–13**	**1.07E+03**	**3.16E–10**	**6.99E+10**	7.28E+14	**1.10E+05**	1.78E+09	**7.75E+14**
Mean	**2.03E–13**	**1.03E+03**	**2.87E–10**	**2.60E+10**	7.28E+14	**4.85E+04**	6.07E+08	**4.26E+14**
Std	**1.78E–14**	**2.26E+01**	**1.38E–11**	**1.47E+10**	**1.51E+05**	3.98E+04	4.09E+08	**1.53E+14**
LS-EDA								
Best	1.03E+07	1.84E+03	2.06E+01	5.95E+11	**7.72E+06**	1.06E+06	2.22E+08	1.52E+17
Median	1.08E+07	1.84E+03	2.14E+01	6.26E+11	**8.49E+06**	1.06E+06	**2.57E+08**	2.09E+17
Worst	1.15E+07	2.17E+03	2.18E+01	7.69E+11	**9.71E+06**	1.06E+06	**3.20E+08**	2.29E+17
Mean	1.08E+07	1.94E+03	2.16E+01	6.48E+11	**8.71E+06**	1.06E+06	**2.61E+08**	2.03E+17
Std	5.71E+05	1.32E+02	2.49E–03	6.51E+10	9.06E+05	**1.20E+03**	**3.72E+07**	3.20E+16

Algorithms	f9	f10	f11	f12	f13	f14	f15	–
DECC-G								
Best	**2.20E+08**	**9.29E+04**	**4.68E+10**	9.80E+02	2.09E+10	1.91E+11	4.63E+07	
Median	**4.17E+08**	**1.19E+07**	**1.60E+11**	1.03E+03	3.36E+10	6.27E+11	6.01E+07	
Worst	**6.55E+08**	**1.73E+07**	7.16E+11	**1.20E+03**	4.64E+10	1.04E+12	7.15E+07	
Mean	**4.27E+08**	**1.10E+07**	2.46E+11	**1.04E+03**	3.42E+10	6.08E+11	6.05E+07	
Std	9.89E+07	4.00E+06	2.03E+11	**5.76E+01**	6.41E+09	2.06E+11	6.45E+06	
LS-EDA								
Best	6.19E+08	9.33E+07	7.61E+10	**9.71E+02**	**1.03E+09**	**2.47E+09**	**6.93E+06**	
Median	6.38E+08	9.41E+07	1.65E+11	**9.89E+02**	**1.45E+09**	**3.17E+09**	**8.25E+06**	
Worst	7.55E+08	9.44E+07	**2.28E+11**	2.49E+03	**2.63E+09**	**4.11E+09**	**8.86E+06**	
Mean	6.58E+08	9.40E+07	**1.69E+11**	1.15E+03	**1.46E+09**	**3.33E+09**	**8.06E+06**	
Std	**6.58E+08**	**2.51E+05**	**3.88E+10**	3.63E+02	**6.10E+08**	**2.01E+11**	**5.61E+05**	

This result is reported by the best, median, worst, mean, and standard deviation of the 25 runs when the functions evolutions counter (#eval) reaches FEs = 3.0e+06. The best results among all methods are highlighted using boldface

$\omega = 0.5$ and $N = 200$. The experimental results are given in Table 10. Please note that we do not perform DECC-G, and its results are directly cited from the CEC2015 site.

Table 10 shows that DECC-G is superior to LS-EDA in terms of accuracy on the fully separable functions f1–f3. As to the reason, LS-EDA tries to capture the principal component, but each variable is separable and projection transformation of LS-EDA may lead to oscillation of the principal component among different variables. In contrast, DECC-G can decompose a high-dimensional problem into some low-dimensional subcomponents and evolve these subcomponents cooperatively.

On the partially separable functions with a separable subcomponent (f4–f7), DECC-G outperforms LS-EDA on f4 and f6, whereas LS-EDA significantly outperforms DECC-G on f5. On f7, DECC-G is better than LS-EDA on the best value, but worse than LS-EDA on the median value, worst value, mean value and standard deviation value. It shows that although DECC-G can find a best solution, it is less stable than LS-EDA. On the partially separable functions without separable subcomponents (f8–f11), their performances show no statistically significant difference.

For the last two groups of functions: overlapping functions (f12–f14) and nonseparable functions f15, LS-EDA performs significantly better than DECC-G. It is clear that the cooperative coevolution strategy of DECC-G is no longer effective for such cases. Because LS-EDA adopts pPCA, it can effectively find the principal component among variables to guide a search process well. Consistent with the previous experiments, we can find LS-EDA is an outstanding candidate well suited to high-dimensional nonseparable problems.

5 Characterization of problem properties by LS-EDA

Similar to EDA-MCC, LS-EDA can look into the structural characteristics of optimization problems. Since LS-EDA

employs the probabilistic model based on a latent subspace whose coordinates are corresponding to principal components, the projections of each original coordinate into the latent subspace give us a chance to reveal problem properties. Only a few studies have paid attention to the advantage of using EDA to obtain further information of problems except for finding a better solution. In Omidvar et al. (2014), the interactions of potential information are reported by spying on the probability model in a discrete EDA framework. In Dong et al. (2013), two strategies in EDA-MCC, i.e., weakly dependent variable identification (WI) and subspace modeling (SM), are used to visually analyze the structure of optimization problems. In EDA-MCC, the number of strongly dependent variables and times of each variable belonging to a dependent subspace in 25 runs are recorded during the evolution to serve as indexes to find the structural characteristics of problems.

Following their method in Dong et al. (2013), we show structural characteristics of optimization problems by visualizing an evolutionary process. For this purpose, we record the number of times for which each dimension variable's projected value on the principal component in every generation is the largest one among all variables. We have:

$$p_{\vec{v}}^{\vec{u}} = |\vec{u}| \cos\theta = \frac{|\vec{u}| \times |\vec{v}| \times \cos\theta}{|\vec{v}|} = \frac{\vec{u} \bullet \vec{v}}{|\vec{v}|} \quad , \tag{22}$$

where θ is the angle between vectors $\vec{u}$ and $\vec{v}$, $\vec{v}$ is the eigenvector corresponding to the first principal component and $\vec{u}$ is given by

$$\vec{u} = \vec{x}_{\text{best}} - \vec{m}, \tag{23}$$

where $\vec{x}_{\text{best}}$ is the best individual and $\vec{m}$ is the central point of the current population. By analyzing these records, problem properties can be discovered. We count the times among total 25 runs for each variable whose projected absolute value on the principal axis is the maximal one out of all the variables during evolution by a matrix R. Each row of R corresponds to a variable, and each column corresponds to one generation. Obviously, its element R_{ij} on the ith row and jth column ranging from 0 to 25 indicates the number of times (out of the 25 runs) of the ith variable whose projected value on the principal axis is the maximal one out of all the variables at generation j. Because it is hard to see clearly for human eyes to visually examine matrix R when dimensionality exceeds 10, we only perform additional 10-D experiments for LS-EDA. The same parameters for 10-D tests are set as those for the previous 50-D and 100-D experiments in Sect. 4. Similar to Dong et al. (2013), we also transform the column (the number of generations) of R into the number of function evaluations (FEs) in all the following figures. The horizontal axis of the graph is also converted to FEs. Due to the space limit, only the results on F1, F6, F8 and F12 are reported in Figs. 6, 7, 8 and 9, respectively.

In all these figures, the meaning of sub-figures is as follows: Sub-figure (a) shows the maximum projected value of the each dimension variable on the principal component. The horizontal coordinate refers to the number of FEs, and the vertical one refers to the variable of an optimization problem. Sub-figure (b) shows the maximum positive correlation and negative correlation times between a variable and the principal component. The result by Eq. (22) may be negative or positive, and negative value means negative correlation, while positive one means positive correlation. The value of the horizontal coordinate above 0 indicates a positive correlation between a variable and the principal component, whereas the value of the horizontal coordinate below 0 indicates a negative correlation. Sub-figure (c) shows the projected value of the each dimension variable on the principal component in the best population.

Figure 6 shows that on separable F1, every dimension variable has the largest projected value in the evolutionary process. It can be interpreted that all variables do not interact with each other. Furthermore, the colors of Fig. 6a, b are more evenly distributed, indicating that all the variables are observed to play identical roles in optimizing F1. Figure 7 clearly shows that the first variable is constantly identified as

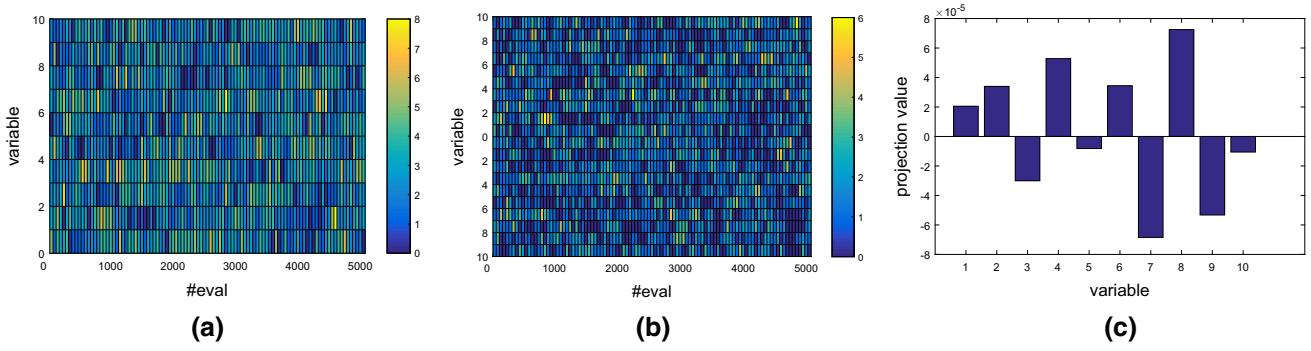

Fig. 6 The problem properties on F1

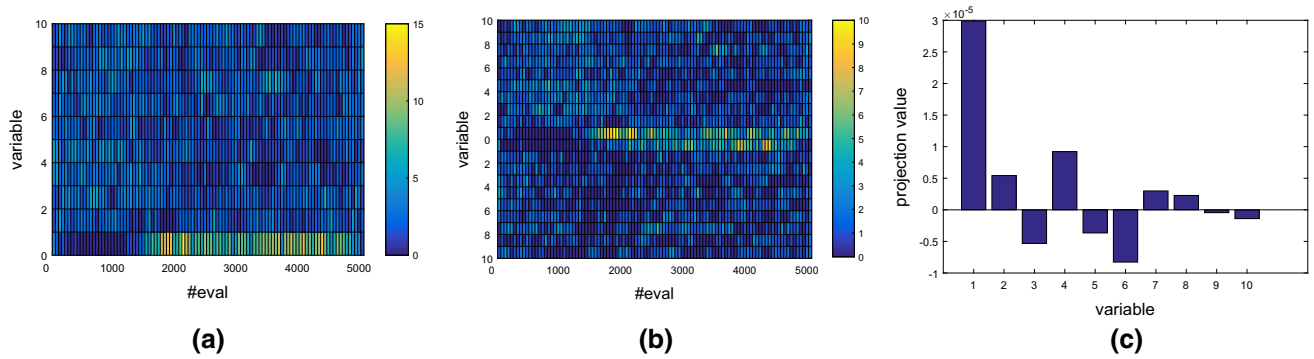

Fig. 7 The problem properties on F6

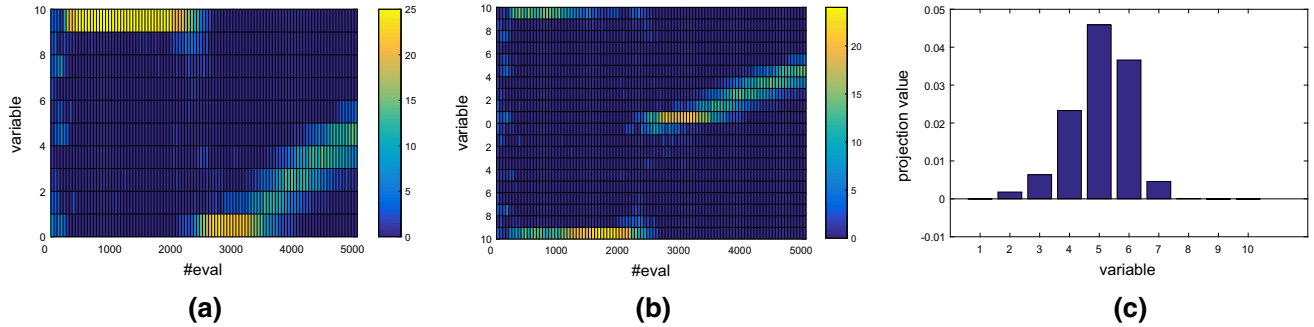

Fig. 8 The problem properties on F8

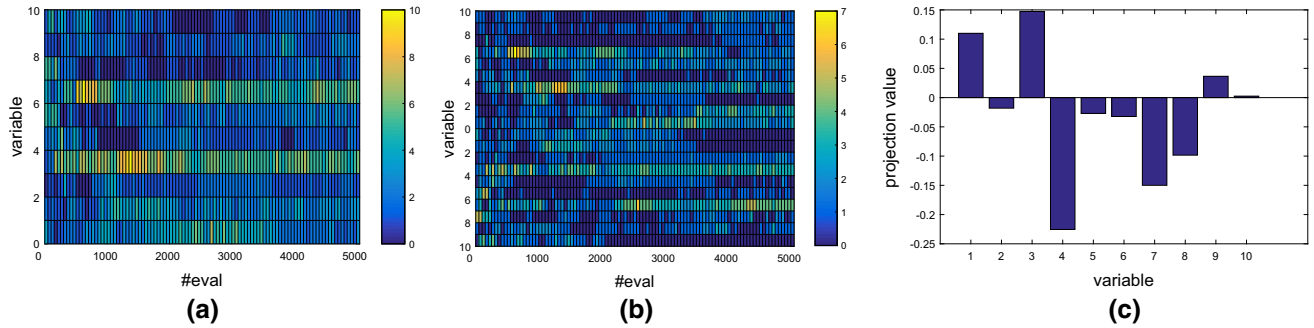

Fig. 9 The problem properties on F12

strongly dependent on the principal component during evolution, and it plays a significant role in this problem.

Figure 8 shows that LS-EDA correctly recognizes the problem structure of shifted Rosenbrock F8. The variable dependency in the problem is a chain-like structure, that is, the first variable determines the second one, the second one determines the third one and so on. This chain-like structure is both reflected in Fig. 8a, b. Figure 8b shows that the negative projected value of the tenth variable on the principal component is the largest one at the beginning, which is consistent with F8. Figure 8c shows the projection value of the fifth and sixth variables in the best population is the largest, which indicates that the fifth and sixth variables are positively correlated with this problem's optimal solution.

Figure 9 shows that the projected values of variables 1, 4 and 7 are always larger than those of the other variables, which means that they play a significant role in solving this problem. But because there are a huge number of local optima in F12, LS-EDA does not identify them as most significant variables. It also explains why LS-EDA fails to solve this problem well. Due to the inefficiency of probabilistic principal component analysis on this function with a huge number of local optima, LS-EDA cannot perform well.

6 Conclusion and future work

Traditional EDAs' performance deteriorates dramatically when the dimensionality of problems grows. Although sev-

eral attempts aim to improve the performance of EDA for large-scale optimization, they fail to fundamentally solve the problem of high computational budget. This paper presents a novel multivariate EDA based on a latent subspace (LS-EDA) to improve the performance while reducing the computational cost to solve LSGOs. By employing a probabilistic PCA model, the proposed method shows significantly better performance over the traditional EDAs on some categories of optimization problems, such as large-scale separable problems and nonseparable problems with only a few local optima. The proposed LS-EDA uses of the EM algorithm reduce the computational complexity, especially for the problems with a high-dimensional space. Experimental results show that the runtime of LS-EDA is lower than those of the other EDAs when the problem dimensionality increases. Similar to EDA-MCC, LS-EDA can also characterize the structure of optimization problems and give some possible clues to provide insight into the properties, especially for black-box optimization problems. Compared with DECC-G, an evolutionary algorithm in a cooperative framework, LS-EDA performs well on the complex problems, such as subcomponent overlapping and nonseparable functions.

The advantages of LS-EDA can be summarized as follows: (1) Its computing budget is lower than the other EDAs, thereby making it more suitable to solve real-world problems; (2) due to EM algorithm, it seldom encounters the problems of unacceptable numerical computation burden, such as finding an irreversible matrix in learning a probabilistic model, and space overflow because of error accumulation; (3) its performance is superior to the traditional EDAs in terms of the accuracy; and (4) its performance is also superior to some evolutionary algorithms with a cooperative framework for highly complex problems.

Based on our experimental results, we also discover some limitations of LS-EDA. First, its performance is worse than evolutionary algorithms with a cooperative framework on simple problems, such as fully separative problems. Secondly, the convergence curve of LS-EDA shows that it is fast at the initial searching stage, but it seems quickly trapped into stagnation. Some open issues are worthy of further study. First, how to determine the number of latent variables to grasp the implicit information of samples needs to be fully tested and analyzed. Moreover, due to their complementary nature between LS-EDA and DECC-G, we should integrate the LS-EDA into a cooperative coevolution framework.

Acknowledgements This study was funded by the NSF of China (Grant Numbers 61170305, 61672024), the NSF of USA (Grant Number CMMI-1162482) and the Key Research Program in Higher Education of Henan (Grant Number 17A520046).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Armananzas R, Saeys Y, Inza I, Garcia-Torres M (2011) Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms. IEEE ACM Trans Comput Biol Bioinform 8(3):760–774

Bielza C, Robles V, Larranaga P (2009) Estimation of distribution algorithms as logistic regression regularizers of microarray classifiers. Methods Inf Med 48(3):236–241

Bishop CM (2006) Pattern recognition and machine learning. Springer, Berlin

Bosman PA, Thierens D (2000) Continuous iterated density estimation evolutionary algorithms within the idea framework. In: Genetic and Evolutionary Computation Conference, page 197200. Optimization Building Using Probabilistic Models OBUPM Workshop GECCO

Bosman PAN, Thierens D (2001) Expanding from discrete to continuous estimation of distribution algorithms: the idea. In: Parallel problem solving from nature—PPSN VI, pp 767–776

Che A, Wu P, Chu F, Zhou M (2015) Improved quantum-inspired evolutionary algorithm for large-size lane reservation. IEEE Trans Syst Man Cybern Syst 45(12):1535–1548

Dong W, Yao X (2008) Unified eigen analysis on multivariate gaussian based estimation of distribution algorithms. Inf Sci 178(15):3000–3023

Dong W, Zhou M (2014) Gaussian classifier-based evolutionary strategy for multimodal optimization. IEEE Trans Neural Netw Learn Syst 25(6):1200–1216

Dong WS, Chen TS, Tino P, Yao X (2013) Scaling up estimation of distribution algorithms for continuous optimization. IEEE Trans Evolut Comput 17(6):797–822. 265RU Times Cited:4 Cited References Count:58

Etxeberria R, Lozano JA (2000) Optimization in continuous domains by learning and simulation of gaussian networks. In: Genetic and Evolutionary Computation Conference

Fogel DB (1996) Evolutionary algorithms in theory and practice, evolutionary programming, genetic algorithms. Oxford University Press, Oxford

Gomes J, Mariano P, Christensen AL (2017) Novelty-driven cooperative coevolution. Evol Comput 25(2):275–307

Guo Z, Zhou MC, Jiang G (2008) Adaptive sensor placement and boundary estimation for monitoring mass objects. IEEE Trans Syst Man Cybern Part B 38(1):222–232

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Hauschild MW, Pelikan M, Sastry K, Goldberg DE (2008) Using previous models to bias structural learning in the hierarchical boa. In: Genetic and Evolutionary Computation Conference, GECCO-2008, pp 415–422

Kabn A, Bootkrajang J, Durrant RJ (2015) Toward large-scale continuous eda: a random matrix theory perspective. Evol Comput 24:255–291

Larranaga P, Lozano JA (2002) Estimation of distribution algorithms: a new tool for evolutionary computation, vol 2. Springer, Berlin

Lin T, Zhang H, Zhang K, Tu Z, Cui N (2016) An adaptive multiobjective estimation of distribution algorithm with a novel Gaussian sampling strategy. Soft Comput 21:6043–6061

Lohn JD, Colombano SP (1999) A circuit representation technique for automated circuit design. IEEE Trans Evol Comput 3(3):205–219

Lu Q, Yao X (2005) Clustering and learning gaussian distribution for continuous optimization. IEEE Trans Syst Man Cybern Part C Appl Rev 35(2):195–204

Mahdavi S, Shiri ME, Rahnamayan S (2015) Metaheuristics in large-scale global continues optimization: a survey. Inf Sci 295:407–428

Mahdavi S, Rahnamayan S, Shiri ME (2016a) Incremental cooperative coevolution for large-scale global optimization. Soft Comput 22:2045–2064

Mahdavi S, Rahnamayan S, Shiri ME (2016b) Multilevel framework for large-scale global optimization. Soft Comput 21:4111–4140

Mishra KM, Gallagher M (2014) A modified screening estimation of distribution algorithm for large-scale continuous optimization. In: Asia-Pacific Conference on Simulated Evolution and Learning. Springer, pp 119–130

Mühlenbein H, Paass G (1996) From recombination of genes to the estimation of distributions I. Binary parameters. In: Parallel Problem Solving from Nature—PPSN IV. Springer, pp 178–187

Omidvar MN, Li X, Yao X (2011) Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. ACM, pp 1115–1122

Omidvar MN, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393

Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) Dg2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942

Pelikan M, Goldberg DE, Lobo FG (2002) A survey of optimization by building and using probabilistic models. Comput Optim Appl 21(1):5–20

Pok P (2005) On the utility of linear transformations for population-based optimization algorithms. In: Preprints of the 16th World Congress of the International Federation of Automatic Control, pp 281–286

Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Parallel problem solving from nature—PPSN III. Springer, pp 249–257

Ray T, Liew KM (2003) Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans Evol Comput 7(4):386–396

Santana R, Mendiburu A, Lozano JA (2012) Structural transfer using EDAs: an application to multi-marker tagging SNP selection. In Evolutionary Computation, pp 1–8

Santana R, Armaanzas R, Bielza C, Larraaga P (2013) Network measures for information extraction in evolutionary algorithms. Int J Comput Intell Syst 6(6):1163–1188

Sanyang ML, Kabn A (2015) Heavy tails with parameter adaptation in random projection based continuous EDA. In: 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 2074–2081

Sebag M, Ducoulombier A (1998) Extending population-based incremental learning to continuous search spaces. In: Parallel Problem Solving from NaturePPSN V. Springer, pp 418–427

Shang YW, Qiu YH (2006) A note on the extended Rosenbrock function. Evol Comput 14(1):119–126

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technological University

Sun J, Zhang Q, Tsang E, Ford J (2003) Hybrid estimation of distribution algorithm for global optimization. Eng Comput 21(1):91–107

Tang K, Li X, Suganthan PN, Yang Z, Weise T (2013) Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization. Nature Inspired Computation & Applications

Wagner M, Auger A, Schoenauer M (2004) EEDA: a new robust estimation of distribution algorithms. Rapport de Recherche (Res. Rep.)RR-5190

Wu FX, Liu LZ, Zhang WJ (2012) Inference of biological S-system using the separable estimation method and the genetic algorithm. IEEE ACM Trans Comput Biol Bioinform 9(4):955–965

Yang XS (2014) Swarm intelligence based algorithms: a critical analysis. Evol Intell 7(1):17–28

Yang Z, Tang K, Yao X (2007) Differential evolution for high-dimensional function optimization. In IEEE Congress on Evolutionary Computation, 2007. CEC 2007, pp 3523–3530

Yang Z, Ke T, Xin Y (2008a) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999

Yang Z, Tang K, Yao X (2008b) Multilevel cooperative coevolution for large scale optimization. In: IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), pp 1663–1670

Yang M, Omidvar MN, Li C, Li X, Cai Z, Kazimipour B, Yao X (2017) Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans Evol Comput 21(4):493–505

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zhou A, Sun J, Zhang Q (2015) An estimation of distribution algorithm with cheap and expensive local search. IEEE Trans Evol Comput 19(6):807–822