
Natural Computing **3:** 77–112, 2004.

Learning probability distributions in continuous evolutionary algorithms – a comparative review

STEFAN KERN[1], SIBYLLE D. MÜLLER[1], NIKOLAUS HANSEN[1], DIRK BÜCHE[1], JIRI OCENASEK[2] and PETROS KOUMOUTSAKOS[1,2,*]
[1]*Institute of Computational Science – ICoS;* [2]*Computational Laboratory – CoLab, Swiss Federal Institute of Technology (ETH), CH-8092 Zürich, Switzerland (*Author for correspondence: E-mail: petros@inf.ethz.ch)*

Abstract. We present a comparative review of Evolutionary Algorithms that generate new population members by sampling a probability distribution constructed during the optimization process. We present a unifying formulation for five such algorithms that enables us to characterize them based on the parametrization of the probability distribution, the learning methodology, and the use of historical information. The algorithms are evaluated on a number of test functions in order to assess their relative strengths and weaknesses. This comparative review helps to identify areas of applicability for the algorithms and to guide future algorithmic developments.

Key words: adaptation, Bayesian optimization, estimation of distribution algorithm, evolution strategy, evolutionary algorithm, learning, probability distribution

1. Introduction

A class of continuous Evolutionary Algorithms (EA) generates new population members by sampling from a probability distribution that is constructed during the optimization process. The probability distribution characterizes the objective function that is being optimized. For this purpose, one can employ machine learning algorithms to effectively exploit the information that is being obtained during the optimization process. When a priori knowledge is available regarding the type of the underlying distribution, a suitable parametrization can lead to fast convergence rates (Rechenberg, 1973; Schwefel, 1995). However, when such knowledge is not available a strategy is needed in order to learn this distribution from the information available by the selected individuals.

The learning of probability distributions in EAs such as Evolution Strategies (ES) and Genetic Algorithms (GA) has received renewed attention in recent years. One of the key concepts in these algorithms involves the identification of correlations between parameters of selected individuals and

the use of these correlations in an effort to accelerate the convergence rate of the algorithms.

In the field of ES, learning of probability distributions has a long history starting from the *1/5th*-success-rule postulated by Rechenberg (1973). This algorithm uses a Gaussian probability distribution to sample new individuals while the variance of the distribution is varied based on past success rates. A number of different learning strategies have been proposed over the years and Beyer and Schwefel (2002) present an overview of several such approaches.

In GAs, a key motivation for the development of model learning mechanisms was the desire not to destroy, by recombination, favorable partial blocks that appear in the variable vector. The basis for introducing learning in GAs was established with the *Population Based Incremental Learning* (PBIL) algorithm proposed by Baluja and Caruana (1995). In PBIL, the recombination operator is replaced by a vector of independent probabilities of each binary variable. This vector of probabilities is used to sample offspring and is adapted in every generation by an update rule. The basic ideas of PBIL were further enhanced by considering, for example, dependencies between variables. The class of GAs that employ learning and sampling of probability distributions is usually referred to as *Estimation of Distribution Algorithms* (EDA) or *Probabilistic Model Building Genetic Algorithms* (PMBGA) (Müehlenbein and Paass, 1996; Pelikan et al., 1999). They try to find adaptively correlations among the building blocks of variables in order to prevent the recombination operator from deteriorating the performance of the algorithm. Larrañaga (2002) presents a summary of EDAs used in continuous domains.

The goal of this paper is to compare how probability distributions are learnt in different continuous EAs. The comparison includes the following algorithms: the $(1 + 1)$-ES with *1/5th*-success-rule (Rechenberg, 1973), the ES with *Cumulative Step Size Adaptation* (CSA-ES) (Ostermeier et al., 1994), the ES with *Covariance Matrix Adaptation* (CMA-ES) (Hansen and Ostermeier, 1996), *Iterated Density Estimation Evolutionary Algorithms* (ID$\mathbb{E}$As) (Bosman and Thierens, 2000a) with Gaussian distributions, and the *Mixed Bayesian Optimization Algorithm* (MBOA) (Ocenasek and Schwarz, 2002). We compare their structural components such as the parametrization of the underlying distribution, their learning methodologies, and the use of historical information. The performance of the algorithms is assessed by direct comparison on a number of unimodal and multimodal test functions.

The remainder of this paper is organized as follows: In Section 2, a common algorithmic structure of the selected EAs is outlined and the different algorithms are described. In Section 3, the algorithms are compared

analytically based on the aspects stated above. The experimental comparison follows in Section 4. Finally, the paper is concluded in Section 5.

2. Description of the algorithms

We consider the application of EAs to the minimization of the *objective function* $f\colon \mathcal{R}^n \to \mathcal{R}, \boldsymbol{x} \mapsto f(\boldsymbol{x})$, where $\boldsymbol{x}$ is the optimization variable and $\mathcal{R}^n$ is the n-dimensional continuous search space. In the following, $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ denotes a population/set of individuals $\boldsymbol{x}_i \in \mathcal{R}^n$. In this section, we outline the common structure of the algorithms and detail their implementation.

2.1. *Common structure of the compared EAs*

The EAs under consideration can be described by a common pseudo-code, which is given in Figure 1. The evolution loop consists mainly of the following operations:

1. selection of the parent population from the base population;
2. (re-)estimation of the probability distribution based on the parent population;
3. sampling and evaluation of the offspring population;
4. replacement of the base population.

Figure 2 illustrates the evolution loop and the populations involved. The populations $\boldsymbol{X}_{\text{base}}$, $\boldsymbol{X}_{\text{parent}}$, and $\boldsymbol{X}_{\text{offspr}}$ have sizes denoted as N_{base}, N_{parent}, and N_{offspr}, respectively.

2.2. *(1 + 1)-ES with* 1/5th-*success-rule*

The $(1+1)$-ES is one of the first and simplest EAs proposed for optimization. Starting from a single parent, $\boldsymbol{x}^{(g)}_{\text{parent}}$, in every generation a single offspring is generated as a realization of a Gaussian distributed random vector $\boldsymbol{z}$:

$$\boldsymbol{x}^{(g+1)}_{\text{offspr}} = \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}^{(g)}_{\text{parent}}, {\sigma^{(g)}}^2 \boldsymbol{I}), \qquad g = 0, 1, 2, \ldots. \tag{1}$$

where the step size $\sigma^{(g)} \in \mathcal{R}_+$ determines the distribution variance at generation g. The base population consists of the two individuals from Equation (1). The individual with the better objective function value is selected to be the parent for the next generation:

$$\boldsymbol{x}^{(g+1)}_{\text{parent}} = \begin{cases} \boldsymbol{x}^{(g+1)}_{\text{offspr}}, & \text{if } f(\boldsymbol{x}^{(g+1)}_{\text{offspr}}) \leq f(\boldsymbol{x}^{(g)}_{\text{parent}}) \\ \boldsymbol{x}^{(g)}_{\text{parent}} & \text{otherwise.} \end{cases} \tag{2}$$

$g := 0$
initialize $\boldsymbol{X}^{(g)}_{\text{base}}, \boldsymbol{f}^{(g)}_{\text{base}}, \mathcal{P}^{(g)}$
while not *terminate* do
  $\boldsymbol{X}^{(g)}_{\text{parent}}, \boldsymbol{r}^{(g)}$:= select $\boldsymbol{X}^{(g)}_{\text{base}}, \boldsymbol{f}^{(g)}_{\text{base}}$
  $\mathcal{P}^{(g+1)}$:=estimate_probability_distr ($\boldsymbol{X}^{(g)}_{\text{parent}}$[, $\boldsymbol{r}^{(g)}$, $\mathcal{P}^{(g)}$])
  $\boldsymbol{X}^{(g+1)}_{\text{offspr}}$:= sample ($\mathcal{P}^{(g+1)}$)
  $\boldsymbol{f}^{(g+1)}_{\text{offspr}}$:= evaluate ($\boldsymbol{X}^{(g+1)}_{\text{offspr}}$)
  $\boldsymbol{X}^{(g+1)}_{\text{base}}, \boldsymbol{f}^{(g+1)}_{\text{base}}$:=
    replace ($\boldsymbol{X}^{(g+1)}_{\text{offspr}}, \boldsymbol{f}^{(g+1)}_{\text{offspr}}$[, $\boldsymbol{X}^{(g)}_{\text{base}}, \boldsymbol{f}^{(g)}_{\text{base}}, \boldsymbol{X}^{(g)}_{\text{parent}}, \boldsymbol{f}^{(g)}_{\text{parent}}$])
  $g := g + 1$
end do

Figure 1. Common algorithm structure of the compared EAs. $\boldsymbol{X}$: Population of individuals (set of search points). $\boldsymbol{f}$: Vector of objective function values according to $\boldsymbol{X}$. $\mathcal{P}$: Probability distribution. $\boldsymbol{r}$: Ranking of population $\boldsymbol{X}_{\text{parent}}$ due to selection. Optional input parameters, not used all algorithms, are put into square brackets.

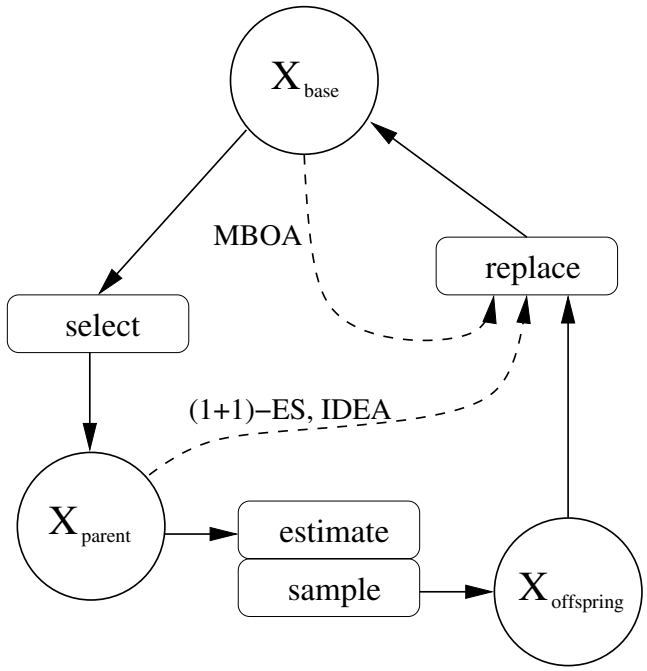

Figure 2. Illustration of the evolution loop. Circles denote populations, boxes denote operators. Arrows pointing to a population indicate their complete replacement. Solid lines apply to all algorithms. Dashed lines apply only to the specified algorithms.

Rechenberg (1973) proposed the so-called *1/5th*-success-rule to adapt the global step size σ based on the rate of successful mutations whereas a mutation is considered successful if $f(\boldsymbol{x}_{\text{offspr}}^{(g+1)}) < f(\boldsymbol{x}_{\text{parent}}^{(g)})$. We propose here an alternative simple implementation of this concept:

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \begin{cases} \alpha & \text{if } f(\boldsymbol{x}_{\text{offspr}}^{(g+1)}) \leq f(\boldsymbol{x}_{\text{parent}}^{(g)}) \\ \alpha^{(-1/4)} & \text{otherwise,} \end{cases} \tag{3}$$

with $\alpha = 2^{1/n}$.[1] This implementation differs from the implementation proposed by Schwefel (1995), in that it accumulates the information about success or failure directly into the step size σ. The new implementation is simpler, as it uses only one strategy parameter α for the change rate. In contrast, the "classical" formulation uses one strategy parameter for the change rate, a second strategy parameter for update frequency, and a third strategy parameter for the averaging time to measure the success rate. These strategy parameters are typically set to 0.817, n, and $10n$, respectively. Compared to the "classical" implementation, we expect the new implementation to achieve the predefined success rate, depending on the choice of α, faster and/or more precisely. Reasonable values for α are between $2^{1/n}$ and 2. The value $2^{1/n}$ resembles the change rate for σ, that has a good performance on the sphere function, $f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$, for all n.

2.3. *ES with Cumulative Step Size Adaptation (CSA-ES)*

The CSA-ES, developed by Ostermeier et al. (1994), adapts the global step size σ by using the path traversed by the parent population over a number of generations.

In the following, the formulation of the $(\mu/\mu_I, \lambda)$-ES with CSA is given, where $\mu \equiv N_{\text{parent}}$ is the number of parents, $\lambda \equiv N_{\text{offspr}} = N_{\text{base}}$ is the number of offspring, and the subscript I denotes intermediate recombination. The offspring $\boldsymbol{x}_k^{(g+1)}$ are sampled from a Gaussian distribution:

$$\boldsymbol{x}_k^{(g+1)} = \boldsymbol{z}_k, \quad \boldsymbol{z}_k \sim \mathcal{N}\left(\langle \boldsymbol{x} \rangle_\mu^{(g)}, {\sigma^{(g)}}^2 \boldsymbol{I}\right), \quad k = 1, \ldots, \lambda, \tag{4}$$

where $\langle \boldsymbol{x} \rangle_\mu^{(g)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{x}_i^{(g)}$ is the result of intermediate recombination of the μ best individuals of generation g, and $\mu < \lambda$. The evolution path $\boldsymbol{p}$ of generation $g+1$ incorporates the mutation steps of the recombined selected individuals and is calculated by

$$\boldsymbol{p}^{(g+1)} = (1-c) \cdot \boldsymbol{p}^{(g)} + \sqrt{c(2-c)} \cdot \frac{\sqrt{\mu}}{\sigma^{(g)}} \left(\langle \boldsymbol{x} \rangle_\mu^{(g+1)} - \langle \boldsymbol{x} \rangle_\mu^{(g)} \right), \tag{5}$$

[1] The exponent $(-1/4)$ corresponds to the success rate of 1/5.

where $1/c$ is the backward time horizon for $\boldsymbol{p}^{(g)}$ typically set to a value between $\sqrt{n}$ and n. The global step size $\sigma^{(g+1)}$ is adapted by comparing the length (Euclidean norm) of the evolution path, $\|\boldsymbol{p}^{(g+1)}\|$, with the expected length of the evolution path under random selection, $\mathrm{E}(\|\mathcal{N}(0,\boldsymbol{I})\|)$:

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \exp\left(\frac{c}{d}\left(\frac{\|\boldsymbol{p}^{(g+1)}\|}{\mathrm{E}\big(\|\mathcal{N}(0,\boldsymbol{I})\|\big)} - 1\right)\right), \tag{6}$$

where $\mathrm{E}(\|\mathcal{N}(0,\boldsymbol{I})\|) = \sqrt{2}\,\Gamma(\frac{n+1}{2})/\,\Gamma(\frac{n}{2}) \approx \sqrt{n}\,(1 - \frac{1}{4n} + \frac{1}{21n^2})$ and $d \approx 1$ is a damping parameter. The strategy parameters c and d may be chosen as follows:

$$c = 10/(n+20), \quad d = \max\left(1, \frac{3\mu}{n+10}\right) + \frac{1}{c}. \tag{7}$$

The initial evolution path is $\boldsymbol{p}^{(0)} = \boldsymbol{0}$.

2.4. *ES with Covariance Matrix Adaptation (CMA-ES)*

Hansen and Ostermeier (1996, 2001) extended the CSA-ES with a derandomized adaptation of the covariance matrix. In the following, a brief summary of the $(\mu/\mu_I, \lambda)$-CMA-ES with additional update of the covariance matrix by a rank-μ matrix (Hansen et al., 2003) is given.

The offspring $\boldsymbol{x}_k^{(g+1)}$ are sampled from a Gaussian distribution:

$$\boldsymbol{x}_k^{(g+1)} = \boldsymbol{z}_k, \quad \boldsymbol{z}_k \sim \mathcal{N}\Big(\langle \boldsymbol{x}\rangle_\mu^{(g)}, {\sigma^{(g)}}^2 \cdot \boldsymbol{C}^{(g)}\Big), \quad k = 1, \ldots, \lambda \tag{8}$$

where $\langle \boldsymbol{x}\rangle_\mu^{(g)} = \frac{1}{\mu}\sum_{i=1}^{\mu} \boldsymbol{x}_i^{(g)}$. The adaptation mechanism of the CMA-ES consists of two parts: (i) the adaptation of the covariance matrix $\boldsymbol{C}^{(g)}$, and (ii) the adaptation of the global step size $\sigma^{(g)}$. The covariance matrix $\boldsymbol{C}^{(g)}$ is adapted by the evolution path $\boldsymbol{p}_c^{(g+1)}$ and by the μ difference vectors between the recent parents and the mean value of the previous parents. The evolution path is computed analogously to that of the CSA:

$$\boldsymbol{p}_c^{(g+1)} = (1 - c_c) \cdot \boldsymbol{p}_c^{(g)} + \sqrt{c_c(2 - c_c)} \cdot \frac{\sqrt{\mu}}{\sigma^{(g)}} \left(\langle \boldsymbol{x}\rangle_\mu^{(g+1)} - \langle \boldsymbol{x}\rangle_\mu^{(g)}\right) \tag{9}$$

$$\boldsymbol{C}^{(g+1)} = (1 - c_{\mathrm{cov}}) \cdot \boldsymbol{C}^{(g)} + c_{\mathrm{cov}} \cdot \left(\frac{1}{\mu}\, \boldsymbol{p}_c^{(g+1)} \left(\boldsymbol{p}_c^{(g+1)}\right)^T + \left(1 - \frac{1}{\mu}\right) \frac{1}{\mu} \sum_{i=1}^{\mu} \frac{1}{{\sigma^{(g)}}^2} \left(\boldsymbol{x}_i^{(g+1)} - \langle \boldsymbol{x}\rangle_\mu^{(g)}\right) \left(\boldsymbol{x}_i^{(g+1)} - \langle \boldsymbol{x}\rangle_\mu^{(g)}\right)^T\right). \tag{10}$$

The last term in Equation (10) is a matrix with rank $\min(\mu, n)$. The strategy parameter $c_{\mathrm{cov}} \in [0, 1[$ determines the rate of change of the covariance

matrix $\boldsymbol{C}$. The adaptation of the global step size $\sigma^{(g+1)}$ is analogous to that of the CSA-ES. In the CMA-ES it is based on a "conjugate" evolution path $\boldsymbol{p}_\sigma^{(g+1)}$:

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\cdot\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\cdot\boldsymbol{B}^{(g)}(\boldsymbol{D}^{(g)})^{-1}(\boldsymbol{B}^{(g)})^{-1}\frac{\sqrt{\mu}}{\sigma^{(g)}}\left(\langle\boldsymbol{x}\rangle_\mu^{(g+1)} - \langle\boldsymbol{x}\rangle_\mu^{(g)}\right). \tag{11}$$

The matrices $\boldsymbol{B}^{(g)}$ and $\boldsymbol{D}^{(g)}$ are obtained through a principal component analysis:

$$\boldsymbol{C}^{(g)} = \boldsymbol{B}^{(g)}\left(\boldsymbol{D}^{(g)}\right)^2\left(\boldsymbol{B}^{(g)}\right)^T, \tag{12}$$

where the columns of $\boldsymbol{B}^{(g)}$ are the normalized eigenvectors of $\boldsymbol{C}^{(g)}$, and $\boldsymbol{D}^{(g)}$ is the diagonal matrix of the square roots of the eigenvalues of $\boldsymbol{C}^{(g)}$. The global step size $\sigma^{(g+1)}$ is determined by

$$\sigma^{(g+1)} = \sigma^{(g)}\cdot\exp\left(\frac{c_\sigma}{d}\left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|}{\mathrm{E}\left(\|\mathcal{N}(0,\boldsymbol{I})\|\right)} - 1\right)\right). \tag{13}$$

The parameters c_σ and c_{cov} control independently the adaptation time scales for the global steps size and the covariance matrix. Default settings for c_c, c_σ, c_{cov}, and d are

$$c_c = \frac{4}{n+4}, \quad c_\sigma = \frac{10}{n+20}, \quad d = \max\left(1, \frac{3\mu}{n+10}\right) + \frac{1}{c_\sigma}, \tag{14}$$

$$c_{\mathrm{cov}} = \frac{1}{\mu}\frac{2}{(n+\sqrt{2})^2} + \left(1-\frac{1}{\mu}\right)\min\left(1, \frac{2\mu-1}{(n+2)^2+\mu}\right). \tag{15}$$

Note that for $\mu \gg n$, d is large and the change of σ is negligible compared to that of $\boldsymbol{C}$. The initial values are $\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{p}_c^{(0)} = \boldsymbol{0}$ and $\boldsymbol{C}^{(0)} = \boldsymbol{I}$.

By choosing $\boldsymbol{x}^{(0)}$ and $\boldsymbol{C}^{(0)}$ appropriately, the CMA-ES is invariant with respect to *any* linear transformation of the search space. By choosing $\boldsymbol{x}^{(0)}$ appropriately, the CMA-ES is invariant with respect to any *orthogonal* linear transformation of the search space, like (1 + 1)-ES and CSA-ES.

In the experimental part of this paper, we use an implementation of the CMA-ES (Hansen et al., 2003) as described above, that additionally uses weighted recombination (Hansen and Ostermeier, 2001). The default settings from the latter paper are used for parent number $\mu = \lambda/2$ and for the recombination weights. Our earlier experiments indicate that the algorithm with weighted recombination and the algorithm with $\mu = \lambda/4$ and equal weights perform similarly.

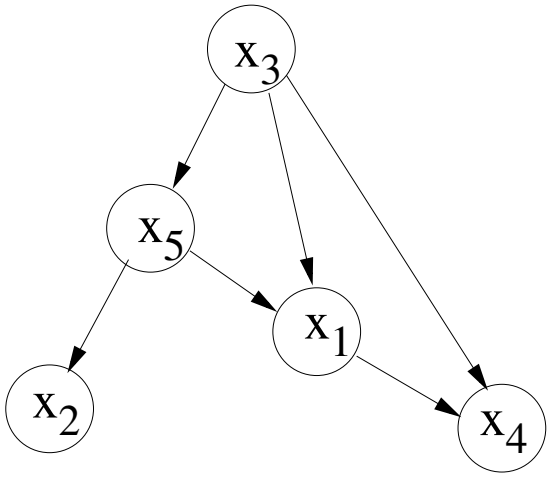

Figure 3. Example of a conditional factorization graph representing the factorization of the joint distribution $P(X_1, \ldots, X_5) = P(X_3) \cdot P(X_5|X_3) \cdot P(X_2|X_5) \cdot P(X_1|X_3, X_5) \cdot P(X_4|X_1, X_3)$.

2.5. *The Iterated Density Estimation Evolutionary Algorithm* (IDEA)

The IDEA framework proposed by Bosman and Thierens (2000a) formalizes EDAs in continuous domains. To estimate the distribution of the parent population, IDEA exploits the fact that every multivariate joint probability distribution can be written as a *conditional factorization*: $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|X_{i+1}, X_{i+2}, \ldots, X_n)$. The probabilistic model $M = (S, \theta)$ of the parent population $\boldsymbol{X}_{\text{parent}}$ is rebuilt in every generation. Here S denotes the structure of the model describing a truncated conditional factorization. Weak variable dependencies are omitted and the maximum number of dependencies per variable may be limited to a fixed number κ. θ is a vector of parameters of the elementary (conditional) probability distributions contained in the factorization. S can be represented by an acyclic directed dependency graph, where arcs indicate conditional dependencies. An example is given in Figure 3.

To learn a suitable conditional factorization, an incremental search algorithm is used starting from an empty graph. In each iteration, an arc is introduced which maximizes a predefined metric J. The graph construction is terminated, when introducing a new arc does not further increase the metric. In this paper, the metric known as the Bayesian Information Criterion (BIC) is used:

$$J_{BIC} = -\ln(L(S|P_M(\boldsymbol{X}_{\text{parent}}))) + \lambda_c \ln(|S|)|\theta|, \tag{16}$$

where $L(S|P_M(\boldsymbol{X}_{\text{parent}}))$ is the likelihood of the model structure S given the distribution of the parent population $\boldsymbol{X}_{\text{parent}}$, and λ_c is a regularization

parameter determining the amount of penalization of the model complexity. An extensive discussion of different search algorithms and metrics for the factorization search in IDEA is given in Bosman and Thierens (2000b).

In this paper, we focus on IDEAs that use Gaussians as elementary probability distribution. Once the factorization is learnt, the parameters of the (conditional) Gaussian distributions are computed from the sample average $\langle \boldsymbol{x} \rangle^{(g)}_{\text{parent}}$ and the sample covariance matrix $\mathbf{C}$ as follows: For each $i = 1, \ldots, n$

$$P(x_i | x_{i+1}, \ldots, x_n) = \mathcal{N}(m_i, \sigma_i)$$

$$\text{where} \quad \begin{cases} \sigma_i &= \dfrac{1}{\sqrt{\boldsymbol{C}^{-1}{}_{(i,i)}}} \\ m_i &= \langle x_i \rangle^{(g)}_{\text{parent}} - \displaystyle\sum_{j=i+1}^{n} \frac{\boldsymbol{C}^{-1}{}_{(j,i)}}{\boldsymbol{C}^{-1}{}_{(i,i)}} \left(x_j - \langle x_j \rangle^{(g)}_{\text{parent}} \right) \end{cases} \tag{17}$$

$$\langle \boldsymbol{x} \rangle^{(g)}_{\text{parent}} = \left(\langle x_1 \rangle^{(g)}_{\text{parent}} \ldots \langle x_n \rangle^{(g)}_{\text{parent}} \right)^T = \frac{1}{N_{\text{parent}}} \sum_{k=1}^{N_{\text{parent}}} \boldsymbol{x}_k^{(g)} \tag{18}$$

$$\boldsymbol{C} = \frac{1}{N_{\text{parent}}} \sum_{k=1}^{N_{\text{parent}}} (\boldsymbol{x}_k^{(g)} - \langle \boldsymbol{x} \rangle^{(g)}_{\text{parent}})(\boldsymbol{x}_k^{(g)} - \langle \boldsymbol{x} \rangle^{(g)}_{\text{parent}})^T. \tag{19}$$

Bosman and Thierens (2001) also present IDEA instances that learn mixtures of Gaussians. In this case, the parents are first clustered, and then for every cluster a probabilistic model is learnt, as outlined above. The use of mixture of Gaussians is claimed to be advantageous for the optimization of non-linear and highly epistatic problems. We investigated IDEAs using Gaussian distributions both with and without clustering.

In every generation, $\tau \cdot N_{\text{base}}$ individuals of the base population are selected as parents. The probabilistic model distribution, $P_M(S, \theta)$, is learnt from the parent population. If P_M is a single peak Gaussian, the offspring are sampled by

$$\boldsymbol{x}_k^{(g+1)} = \boldsymbol{z}_k, \quad \boldsymbol{z}_k \sim \prod_i \mathcal{N}(m_i, \sigma_i), \quad k = 1, \ldots, N_{\text{offspr}}. \tag{20}$$

The base population of the next generation is obtained by merging the offspring population and the parent population of the current generation.

The strategy parameters are set as follows. Population sizes $N_{\text{parent}} = \tau N_{\text{base}}$, $N_{\text{offspr}} = (1 - \tau) N_{\text{base}}$, and the fraction $\tau = 0.3$. The regularization parameter for penalizing the model complexity is $\lambda_c = 0.5$, and the maximum number of dependencies is $\kappa = n - 1$.

2.6. *The Mixed Bayesian Optimization Algorithm (MBOA)*

In MBOA (Ocenasek and Schwarz, 2002), a Bayesian network with local structures in the form of decision trees captures the mutual dependencies among the parent individuals. The first EDA employing the Bayesian network model with decision trees was the hierarchical Bayesian Optimization Algorithm (hBOA) (Pelikan et al., 2000). MBOA is an extension of hBOA from binary to continuous domains. In fact, MBOA is able to deal with discrete and continuous parameters simultaneously, but in this paper we focus on continuous parameters only.

In every generation, the parent population $\boldsymbol{X}_{\text{parent}}$ of size $N_{\text{parent}} = \tau \cdot N_{\text{base}}$ is selected from the base population using tournament selection. Then the probability distribution of $\boldsymbol{X}_{\text{parent}}$ is estimated and N_{offspr} offspring are sampled. The offspring population is used to replace part of the base population. For effective diversity preservation, restricted tournament replacement is used (Pelikan and Goldberg, 2001).

The probabilistic model $M = (\boldsymbol{T}, \theta)$ of the parent population $\boldsymbol{X}_{\text{parent}}$ is rebuild in every generation. $\boldsymbol{T} = \{T_1, \ldots, T_n\}$ is a set of decision trees defining the structural part of the model whereas θ are the quantitative parameters of the model. Each decision tree T_i defines the conditional distribution $P(X_i|\Omega_i)$ of the variable X_i, $i = 1, \ldots, n$. Domain Ω_i denotes the subspace spanned by the variables that affect the value of X_i. The subspace is chosen with regard to all previously generated trees, such that no bidirectional dependencies occur.

To define split nodes in the decision tree T_i, a variable and a split boundary are chosen using Bayesian-Dirichlet metrics. The split nodes hierarchically decompose Ω_i, the domain of $P(X_i|\Omega_i)$, into rectangular axis-parallel partitions, Ω_{ij}, $j = 1, 2, \ldots$, that correspond to the leaves of the decision tree. In each leaf, X_i is approximated by a univariate probability density function using Gaussian kernels (also known as Parzen window (Parzen, 1962)). Let $\Omega_{ij} \in \Omega_i$ denote the partition that traverses to the j-th leaf of T_i. Consider all parent individuals that traverse to Ω_{ij}, and let the set $\{x_i\}_j$ denote their realizations of variable X_i. Then the Gaussian kernel distribution in the j-th leaf can be expressed as:

$$P(X_i|\omega_i \in \Omega_{ij}) = \frac{1}{|\{x_i\}_j|} \sum_{m \in \{x_i\}_j} \mathcal{N}(m, \sigma_{ij}^2) \tag{21}$$

All the kernels in the same leaf have the same height $1/|\{x_i\}_j|$ and the same width σ_{ij}. In our experiments, we set

$$\sigma_{ij} = \frac{\max\{x_i\}_j - \min\{x_i\}_j}{|\{x_i\}_j| - 1}, \tag{22}$$

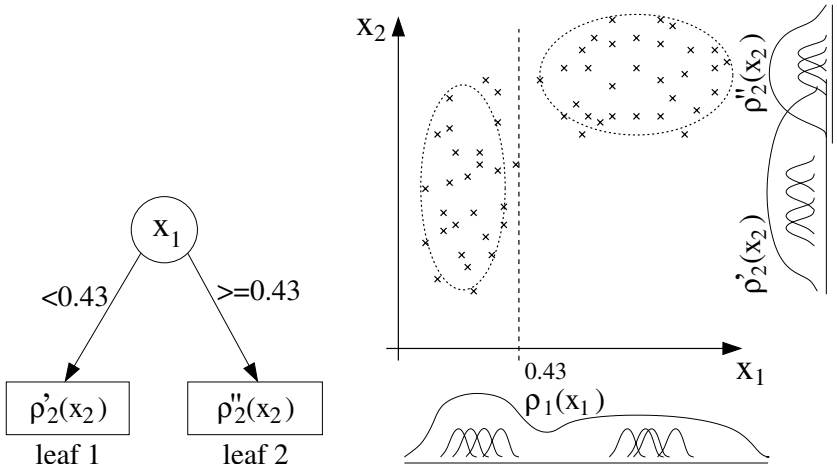

Figure 4. Example of the structure of a two dimensional joint probability distribution $P(X_1, X_2)$ in MBOA. $P(X_1, X_2)$ is factorized as $P(X_1, X_2) = P(X_1) \cdot P(X_2|X_1)$. $P(X_1)$ is described by the density function $\rho_1(x_1)$ and $P(X_2|X_1)$ by the density function $\rho_2(x_2|x_1)$, with $\rho_2(x_2|x_1) = \rho'_2(x_2)$ if $x_1 < 0.43$ and $\rho_2(x_2|x_1) = \rho''_2(x_2)$ if $x_1 \geq 0.43$.

Figure 4 illustrates the structure of learnt distributions in MBOA on a two dimensional example.

The offspring population $\boldsymbol{X}_{\text{offspr}}$ is sampled from the estimated model

$$\boldsymbol{x}_k^{(g+1)} = \boldsymbol{z}_k, \quad \boldsymbol{z}_k \sim \prod_i P(X_i|\Omega_i), \quad k = 1, \ldots, N_{\text{offspr}}. \tag{23}$$

N_{offspr} is set to half of the base population size N_{base}, and the fraction of $\boldsymbol{X}_{\text{base}}$ selected as parent population is set to $\tau = 0.5$.

3. Structural comparison

We compare the algorithms by highlighting their similarities in the type and the parametrization of the employed probability distributions. We also distinguish their differences based on their learning strategies and the exploitation of past information available during the optimization process.

3.1. *The probability distribution and its parametrization*

In the EAs presented herein the employed probability distributions are composed of a single or multiple Gaussian distributions. The Gaussian distributions are characterized by their mean and the covariance matrix, as $\mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$.

Both the (1 + 1)-ES and the CSA-ES learn an isotropic Gaussian ($\boldsymbol{C} = \sigma^2\boldsymbol{I}$), while the CMA-ES and ID$\mathbb{E}$A employ arbitrary Gaussian distributions. If clustering is applied in ID$\mathbb{E}$A, a mixture of arbitrary Gaussian distributions is learnt. MBOA operates with a Gaussian kernel distribution, defined on partitions of the search space. The kernels involve a diagonal covariance matrix that is constant within each partition.

The algorithms are distinguished in the way they determine the parameters of their Gaussian distributions. The ESs use a *fixed* parametrization as determined by the mean vector and the covariance matrix. In contrast, the parametrizations of the distributions in ID$\mathbb{E}$A and MBOA change from generation to generation as the model structure is being rebuilt. In ID$\mathbb{E}$A, the model structure is a conditional factorization graph that is changing in every generation, thus resulting in different parametrizations of the corresponding Gaussian distributions. Similarly, in MBOA, the decision trees that constitute the model structure are re-estimated in each generation resulting as well in changes in the associated Gaussian kernels.

3.2. *Statistical measures used in the learning process*

The probability distributions are being learned by employing statistical measures based on

- the position of individuals in the search space and
- relative rank information among different individuals as determined by the values of the objective function.

Note that using rank information as opposed to function values themselves, has the advantages that it makes the algorithms (i) invariant to any monotonous transformation of the objective function and (ii) insensitive to small-scale perturbations of the objective function. On the other hand, not using the objective function values results in a larger number of evaluations in the case of smooth unimodal objective functions.

In the (1 + 1)-ES with *1/5th*-success-rule, the only statistical measure is the success rate that is used to adapt the global step size. In the CSA-ES and CMA-ES, the mean of the population is used in order to compute the evolution path $\boldsymbol{p}$. The comparison of this evolution path with an evolution path under random selection is used to determine the global step size σ. This comparison is an indirect measure of the correlation of the subsequently selected mutation steps. To adapt the covariance matrix $\boldsymbol{C}$ of the underlying probability distribution, the CMA-ES uses the covariance matrix of the evolution path $\boldsymbol{p}_c$ which has rank one, and the covariance matrix of the parent individuals which has rank $\min(\mu, n)$. To compute the latter covariance matrix, the mean of the previous distribution is used as reference point.

This is in contrast to the computation of the empirical covariance matrix using the sample mean as reference point.

In IDEA, the factorization search and the estimation of the parameters of the Gaussian distributions is based on the mean and the empirical covariance matrix of the selected individuals. The Bayesian Information Criterion (BIC) is used to measure the likelihood of candidate model structures regularized by a term that penalizes model complexity. Once the model structure is learnt, the distribution parameters have to be estimated based on the parents. MBOA uses a measure derived from the Bayesian-Dirichlet metrics to split the search space into axis-parallel partitions. The span of the parents within the partitions is used to estimate the variances.

3.3. *Learning strategies and use of historical information*

The algorithms compared in this paper use two different approaches to learn the probability distribution. The ESs *incrementally update* their distributions in every generation using the new information provided by the offspring. In contrast, IDEA and MBOA *rebuild* the probability distribution in every generation. Referring to Figure 1, the latter algorithms do not require the argument $\mathcal{P}$ for the `estimate_probability_distr` procedure.

The considered EAs use two different mechanisms for preserving and reusing historical information:

- elitist population strategy and
- generation-varying parameters which are modified by deterministic update rules.

The elitist strategy affects the replacement procedure and helps to preserve former parent solutions in the population. Only elitist strategies feed back the parent population or the entire base population to the replacement operator as shown in Figure 2. Additional generation-varying parameters involve distribution parameters of $\mathcal{P}$ or additional internal state parameters attached to $\mathcal{P}$. These parameters are summarized in Table 1.

In this framework the $(1 + 1)$-ES uses an elitist population strategy and a single distribution parameter, the global step size σ. Internal state parameters are not used. The CSA-ES and CMA-ES do not use the elitist population strategy. Both algorithms have internal state parameters and distribution parameters that are being modified by deterministic update rules. The CSA-ES uses the evolution path $\boldsymbol{p}$, which is an internal state parameter, to adapt the global step size σ. In the CMA-ES, two different evolution paths $\boldsymbol{p}_c$ and $\boldsymbol{p}_\sigma$ are used to update the covariance matrix and the overall variance, respectively.

Both IDEA and MBOA use an elitist population strategy. They use neither internal state parameters nor distribution parameters as information is provided solely by the population itself.

Table 1. Generation-varying parameters of the algorithms. Initialization is given for parameters with constant initialization. Algorithms not listed do not have generation-varying parameters

	Distribution parameters	Internal state parameters
$(1+1)$-ES	$\sigma \in \mathcal{R}_+$	
CSA-ES	$\sigma \in \mathcal{R}_+$	$\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{0}$
CMA-ES	$\sigma \in \mathcal{R}_+, \boldsymbol{C}^{(0)} = \boldsymbol{I}$	$\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{0}, \boldsymbol{p}_c^{(0)} = \boldsymbol{0}$

Table 2. Strategy parameters of the algorithms

	N_{parent}	N_{offspr}	Other
$(1+1)$-ES	1	1	$\alpha = 2^{1/n}$, success rate 1/5
CSA-ES	$\mu = N_{\text{base}}/2$	$\lambda = N_{\text{base}}$	$c = \frac{10}{n+20}$, $d = \max\left(1, \frac{3\mu}{n+10}\right) + \frac{1}{c}$
CMA-ES	$\mu = N_{\text{base}}/2$	$\lambda = N_{\text{base}}$	$c_\sigma = \frac{10}{n+20}$, $d = \max\left(1, \frac{3\mu}{n+10}\right) + \frac{1}{c_\sigma}$ $c_c = \frac{4}{n+4}$, $c_{\text{cov}} = \frac{1}{\mu}\frac{2}{(n+\sqrt{2})^2}$ $+ \left(1 - \frac{1}{\mu}\right) \min\left(1, \frac{2\mu-1}{(n+2)^2+\mu}\right)$
IDEA	τN_{base}	$(1-\tau)N_{\text{base}}$	$\tau = 0.3$, $\lambda_c = 0.5$, $\kappa = n-1$ number of clusters threshold for adding a cluster
MBOA	τN_{base}	$N_{\text{base}}/2$	$\tau = 0.5$, window size 5%

3.4. *Strategy parameters*

The implementation of the considered EAs involves the specification of several strategy parameters that can, for example, be determined via a priori knowledge of the objective function at hand or by extensive experimentation. A common strategy parameter is the size of the base population N_{base}. Further strategy parameters are summarized in Table 2.

The $(1+1)$-ES as formulated in Subsection 2.2 employs only one strategy parameter, α, that controls the rate of change of the global step size. In the CSA-ES, the rate of change of the global step size is controlled by the damping factor d. The life span of the information accumulated in the evolution path is controlled by the accumulation parameter c. In the CMA-ES, additionally to c_σ, d (which have the same meaning as in the CSA-ES), and c_c, the parameter c_{cov} is used to control the learning rate of the covariance

matrix $\boldsymbol{C}$. In this way, the distribution estimation task is detached from the choice of the population size. The number of parents is set to $N_{\text{parent}} = N_{\text{base}}/2$ and the number of offspring is set to $N_{\text{offspr}} = N_{\text{base}}$. N_{parent} does not reflect the time selection pressure, because the parents are weighted for the estimation procedure.

The IDEA employs a regularization parameter, λ_c, that determines the amount of penalization for the model complexity in the BIC-metric used for the factorization search. If clustering is used, IDEA additionally uses two strategy parameters in the leader clustering algorithm namely a threshold and a maximum number of clusters. The maximum number of parent variables for each variable is limited by κ. The fraction τ of selected parent individuals is a further strategy parameter. Similar to IDEA, the MBOA algorithm employs a strategy parameter that penalizes complexity. This penalty parameter is used in the metric for building decision trees. By appropriately choosing this parameter, one can avoid to overfit the model. In the restricted tournament replacement, the window size determines the number of individuals any offspring has to compete with.

4. Experimental comparison

4.1. *Test functions, algorithm implementation, and testing procedure*

The test functions used in this paper are summarized in Table 3. Unless otherwise noted experiments are conducted for dimension $n = 10$. All test functions are minimized except for the plane and the diagonal plane which are maximized. We study functions of the variables $\boldsymbol{x}$ and functions of the transformed variables $\boldsymbol{y}$. The transformation reads $\boldsymbol{y} := \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{A} = [\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n]^T$ implements an angle-preserving (i.e. orthogonal) linear transformation of $\boldsymbol{x}$, and each $\boldsymbol{o}_i$ is uniformly distributed on the unit hypersphere surface (Hansen and Ostermeier, 2001). For $\boldsymbol{A} = \boldsymbol{I}$ (without transformation), the Rosenbrock function is the only nonseparable function of the test suite. For $\boldsymbol{A} \neq \boldsymbol{I}$, the respective functions become nonseparable. The functions f_{elli}, f_{cigar}, f_{tablet}, $f_{\text{Rastrigin}}$, and $f_{\text{Rastrigin10}}$ are tested both without and with transformation. The transformed functions are also referred to as rotated functions. To test the performance of the algorithms, we consider 20 different runs for each function. To optimize a transformed function, we use ten different transformations and for each of them we perform two runs.

We performed our experiments with implementations provided by the authors of the algorithms, with the exception of the $(1 + 1)$-ES where a MATLAB implementation of the equations in Subsection 2.2 was used. The MATLAB code of the CMA-ES supports large population sizes and uses

Table 3. Test functions and initialization intervals (coordinate-wise), where $\boldsymbol{y}: = [\boldsymbol{o}_1 \ldots, \boldsymbol{o}_n]^T \boldsymbol{x}$, i.e. $y_i = \boldsymbol{o}_i^T \boldsymbol{x}$, see text

Name	Function	Init
Plane	$f_{\text{plane}}(\boldsymbol{x}) = x_1$	$[0.5, 1.5]^n$
Diagonal plane	$f_{\text{planediag}}(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^n x_i$	$[0.5, 1.5]^n$
Sphere	$f_{\text{sphere}}(\boldsymbol{x}) = \sum_{i=1}^n x_i^2$	$[-3, 7]^n$
Ellipsoid	$f_{\text{elli}}(\boldsymbol{y}:) = \sum_{i=1}^n \left(100^{\frac{i-1}{n-1}} y_i\right)^2$	$[-3, 7]^n$
Cigar	$f_{\text{cigar}}(\boldsymbol{y}:) = y_1^2 + 10^4 \sum_{i=2}^n y_i^2$	$[-3, 7]^n$
Tablet	$f_{\text{tablet}}(\boldsymbol{y}:) = 10^4 y_1^2 + \sum_{i=2}^n y_i^2$	$[-3, 7]^n$
Rosenbrock	$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	$[-5, 5]^n$
Rastrigin	$f_{\text{Rastrigin}}(\boldsymbol{y}:) = 10n + \sum_{i=1}^n \left(y_i^2 - 10\cos(2\pi y_i)\right)$	$[-3, 7]^n$
Scaled Rastrigin	$f_{\text{Rastrigin10}}(\boldsymbol{y}:) = 10n + \sum_{i=1}^n \left((10^{\frac{i-1}{n-1}} y_i)^2 - 10\cos(2\pi 10^{\frac{i-1}{n-1}} y_i)\right)$	$[-3, 7]^n$

weighted recombination (compare end of Subsection 2.4). The same code is used for the CSA-ES by setting $c_{\text{cov}} = 0$. The IDEA implementation was provided in C. It uses Gaussians as elementary probability distributions, the BIC-penalized negative log-likelihood as metric in the factorization search, and the leader algorithm for the clustering. For MBOA, a C implementation was used. In order to facilitate the reproducibility of this investigation the source code of all algorithms, as they were used, is available under http://www.icos.ethz.ch/software/optimization/nc/.

For all test functions, our performance criterion was the number of function evaluations to reach a certain function value f_{stop}. This number depends strongly on the initialization, on the value of f_{stop}, and on the size of the population. We used the following experimental setup:

Initialization: The initial intervals are given in Table 3. For MBOA and IDEA, the initial population distribution and the initial distribution $\mathcal{P}$ (see Figure 1) is a uniform distribution on the initial interval. For (1 + 1)-ES, CSA-ES, and CMA-ES, the initial distribution is set to $\mathcal{P} = \mathcal{N}(\boldsymbol{m}, \sigma^2 \boldsymbol{I})$, where $\boldsymbol{m}$ is uniformly sampled from the initial interval and σ equals half of the interval size.

In preliminary experiments, we varied the size and location of the initial interval, and observed the following: For IDEA, it is essential that the starting region contains the optimum, otherwise IDEA does not find the optimum. For MBOA, it is beneficial that the starting region contains

the optimum. For (1 + 1)-ES, CSA-ES, and CMA-ES the performance is insensitive against the placement of the starting region on unimodal test functions.

Termination criterion: Each run is stopped when the function value is smaller than $f_{\text{stop}} = 10^{-10}$ (minimization), except for the plane and the diagonal plane where $f_{\text{stop}} = 10^{10}$ (maximization). Additionally, the run is stopped if a certain number of function evaluations was exceeded or the optimization converged prematurely. Premature convergence is indicated by a vanishing variance of the learnt model distribution before f_{stop} is reached. Experiments in which not all of the 20 runs reached f_{stop} are marked in Table 4.

Performance results are highly sensitive to the chosen value of f_{stop}. Therefore, for all simulations we supply graphs in which the performance for function values $\geq f_{\text{stop}}$ can be seen.

Population size: For the (1 + 1)-ES we performed 20 runs. For the remaining algorithms, we used as population size the testing sequence $N_{\text{base}} = [10, 20, 50, 100, 200, 400, 800, 1600, 3200]$ performing 20 runs for each population size. The number of function evaluations to reach f_{stop} are given for the smallest population size, where all 20 runs reached f_{stop}. If f_{stop} was not reached in all 20 runs, we report results for that N_{base} with the most successful runs. If no single run reached f_{stop}, we report the best achieved function value. For IDEA using 10 clusters, the upper limit for N_{base} was 16 000 on f_{Rosen}, $f_{\text{Rastrigin}}$, and $f_{\text{Rastrigin10}}$.

The best population size strongly depends on the algorithm *and on the test function*. Therefore, the procedure to increase the population size could also be valuable in practice. But, there is a clear disadvantage in this method: Results achieved with different population sizes are not necessarily comparable.

Strategy parameters: Beyond testing different population sizes as outlined before, and testing different cluster sizes in IDEA, we did not test different strategy parameter systematically. Instead, we use strategy parameters suggested by the authors of the algorithms. Their values are given in Subsections 2.2–2.6.

IDEA with clustering was only applied to f_{Rosen}, $f_{\text{Rastrigin}}$, and $f_{\text{Rastrigin10}}$, where different numbers of clusters with varying threshold values were tested (see Subsections 4.3.6, 4.4.1, and 4.4.2).

Figures 6–18 show the function values over the number of function evaluations for the 10-dimensional test functions described in Table 3. Bold lines

are the median of the function values from 20 runs. Thin lines show the curves of the minimum and maximum function value. The five symbols per each measurement represent minimum, 25-percentile, median, 75-percentile, and maximum function values. The population sizes are given in the caption. The performance ratios of the algorithms on all test functions are summarized in Table 4.

After studying the runtime of the algorithms in Subsection 4.2, we start our discussion on unimodal test functions in Subsection 4.3, and continue with multimodal test functions in Subsection 4.4.

4.2. *Runtime study*

The CMA-ES, IDEA, and MBOA are computationally expensive algorithms, compared to the $(1+1)$-ES and the CSA-ES. To assess the computational cost of the three expensive algorithms, we measure CPU time consumption on an Intel Pentium 4, 2.5 GHz processor on the Rosenbrock function f_{Rosen}. The time to compute the function value of f_{Rosen} is small compared with the time to perform the algorithms, e.g., decomposing the covariance matrix in CMA-ES, or the learning parts in MBOA or IDEA. To make a fair comparison, all three algorithms should be tested using the same programming language. We therefore use a C-implementation of the CMA-ES.[2]

For the dimensions $n = 5, 10, 20, 40$, we measure the total CPU time needed by the algorithm and divide it by the number of function evaluations during that time. In Figure 5, CPU times per function evaluation are plotted, where $N_{\text{base}} = 50$ and 400. The CPU times are between $2 \cdot 10^{-8} n^2$ and $2 \cdot 10^{-6} n^2$ seconds per function evaluation.

When comparing algorithms, we are interested in the scale-up of the algorithms. In our case, scale-up is limited by the computational cost of the experiments. Our experiments, presented for dimension $n = 10$, took more than 2 months of CPU time on Intel Pentium 4 processors with 2.5 and 3 GHz. How long would it take to do all experiments for larger dimensions? We did single scale-up experiments where the number of function evaluations to reach f_{stop} was measured for larger dimensions. Increasing the dimension by a factor of two (i) increases the number of function evaluations to reach f_{stop} by a factor of at least two to four, and (ii) increases the CPU time per function evaluation by a factor of four (Figure 5), resulting in a total CPU time consumption increase by a factor of about ten. This means, to get the complete picture for $n = 20$ and 40 would take about 2 and 20 years of CPU time, respectively.

[2] In small dimensions, the C-implementation is faster than the MATLAB-implementation, but for $n > 30$ the runtimes become similar.

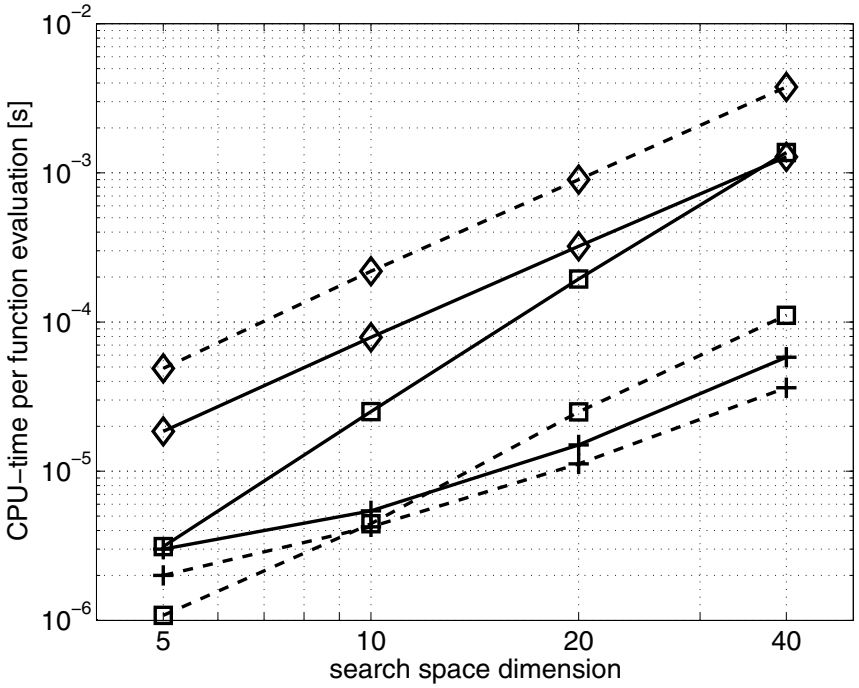

Figure 5. CPU times per function evaluation in seconds versus search space dimension n on a Pentium 4, 2.5 GHz, processor. CMA-ES (+); IDEA (□); MBOA (◇). $N_{\text{base}} = 50$ (solid), $N_{\text{base}} = 400$ (dashed).

4.3. *Linear and unimodal test functions*

4.3.1. *Plane function*

The plane function tests the ability of the algorithms to enlarge the overall population variance as within a small enough neighborhood, a linear function is a reasonable approximation for any smooth function. Therefore, f_{plane} is a good test case for a situation where the population variance is (far) too small.

The performance on f_{plane} is similar for the $(1 + 1)$-ES, CSA-ES, and CMA-ES, see Table 4 and Figure 6. These algorithms increase the variance fast. The slope for the $(1 + 1)$-ES is defined by α (see Equation 3) and is somewhat arbitrary.[3] While the ESs need around 10^3 function evaluations to reach f_{stop}, MBOA needs almost 10^6 function evaluations. IDEA does not reach values better than 2 after 10^6 function evaluations. IDEA not only fails

[3] Here, the choice of α has a large influence on the performance of the $(1+1)$-ES. Choosing $\alpha = 2$ instead of $\alpha = 2^{\frac{1}{n}}$ makes the $(1 + 1)$-ES n times faster. Choosing $\alpha \gg 2$ enhances its performance on f_{plane}, but will have deteriorating effects on most other functions.

Table 4. Performance ratios where $n = 10$. The performance ratio of an algorithm on a given function is defined as follows: The number of function evaluations to reach f_{stop} with the median of 20 runs divided by the same measure for the fastest algorithm on this function. For a ratio of 1 (fastest algorithm), we put the number of function evaluations in round brackets. The * symbol indicates that the algorithm did not reach f_{stop} in every run. Square brackets include the obtained function values in cases when the median run did not reach f_{stop}. The ∞ symbol indicates that the distribution variance decreases below 10^{-15} before reaching f_{stop}. Besides the number of function evaluations, all numbers are given with precision of two digits

Function	$(1+1)$-ES	CSA-ES	CMA-ES	ID$\mathbb{E}$A	MBOA
Plane	**1.0** (790)	1.6	1.4	∞ [1.63]	1100
Diagonal plane	**1.0** (836)	1.5	1.3	∞ [2.76]	49
Sphere	**1.0** (1370)	1.6	1.3	5.0	48
Ellipsoid	66	110	**1.0** (4450)	1.6	14
Cigar	610	800	**1.0** (3840)	4.6	12
Tablet	27	38	**1.0** (4380)	1.7	14
Rotated ellipsoid	64	110	**1.0** (4490)	13	1800 [1.1E-6]
Rotated cigar	600	800	**1.0** (3840)	38	2100 [1.2E-1]
Rotated tablet	25	36	**1.0** (4400)	6.8	910 [4.7E-6]
Rosenbrock	*51	180	**1.0** (7190)	210 [7.5]	1100 [3.3E-3]
Rastrigin	∞ [3.9E1]	14	**1.0** (64000)	20	3.6
Scaled Rastrigin	∞ [1.1E2]	110	**1.0** (40400)	30	6.0
Rotated Rastrigin	∞ [4.4E1]	14	**1.0** (64000)	*38	78 [3.0]
Rotated scaled Rastrigin	∞ [1.2E2]	65	**1.0** (67200)	*120	74 [8.1]

to increase the overall distribution variance, but even worse, the distribution variance converges to zero on a linear function.

The results for the diagonal plane, $f_{\text{planediag}}$, do not differ fundamentally from those on f_{plane}, see Figure 7. Only MBOA is around 30 times faster than on f_{plane}, requiring a much smaller population size. In contrast to f_{plane}, independent sampling (as substitute for recombination) can improve the offspring compared to the parents on $f_{\text{planediag}}$.

The reason for the high number of function evaluations needed in MBOA and for the failure of ID$\mathbb{E}$A is the (missing) concept for overall variance control. While the ESs adapt their global step sizes, MBOA and ID$\mathbb{E}$A do not exhibit an adaptation of the overall distribution width. Estimating the empirical variance of the selected population decreases the overall variance

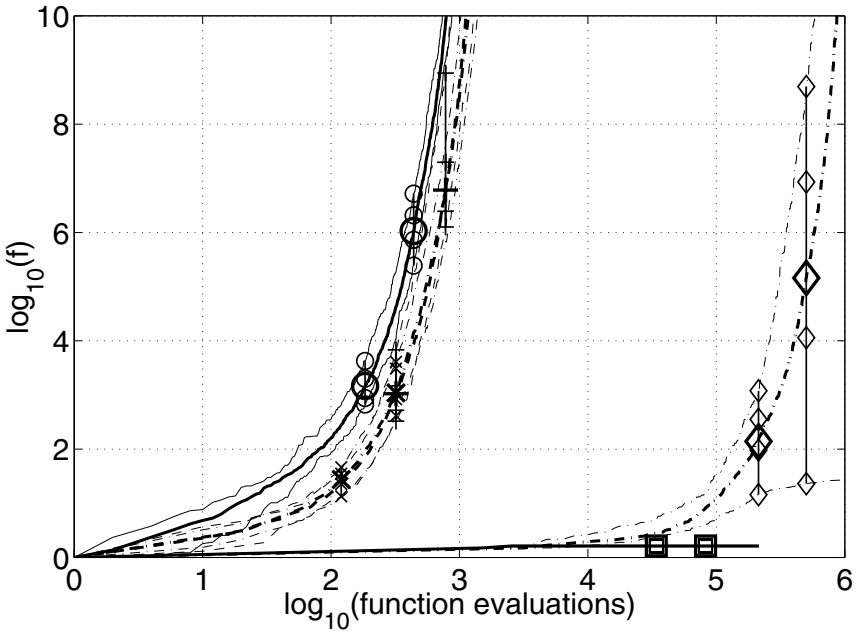

Figure 6. Plane function. (1 + 1)-ES ('—◯—'). Population sizes: $N_{\text{base}} = 10$ for CSA-ES ('$-\cdot-\times-\cdot-$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 1600$ for IDEA ('—□—'); $N_{\text{base}} = 3200$ for MBOA ('$-\cdot-\diamond-\cdot-$').

of IDEA in the generation sequence. Without an opposite mechanism, this decrease easily leads to premature convergence.

4.3.2. *Sphere function*

On f_{sphere}, (1 + 1)-ES, CMA-ES, and CSA-ES perform as well as one would expect on a well-scaled function. They need three to five times less function evaluations than IDEA to reach f_{stop}, see Figure 8. IDEA performs reasonably well for the given population size. MBOA performs almost 50 times slower than the (1 + 1)-ES. The reason for the slow convergence is the restricted tournament replacement which prevents the overall variance from fast shrinking. Without restricted tournament replacement, the variance shrinks too fast and premature convergence occurs. Like on f_{plane}, the (missing) concept for estimating the overall distribution variance plays the key role for the performance on f_{sphere}.

4.3.3. *Ellipsoid function*

The CMA-ES and IDEA are the fastest methods on the axis-parallel f_{elli}, followed by MBOA, (1 + 1)-ES, and CSA-ES, see Figure 9. Up to a function value of 10^{-4} and 10^{0}, respectively, IDEA and MBOA outperform the CMA-

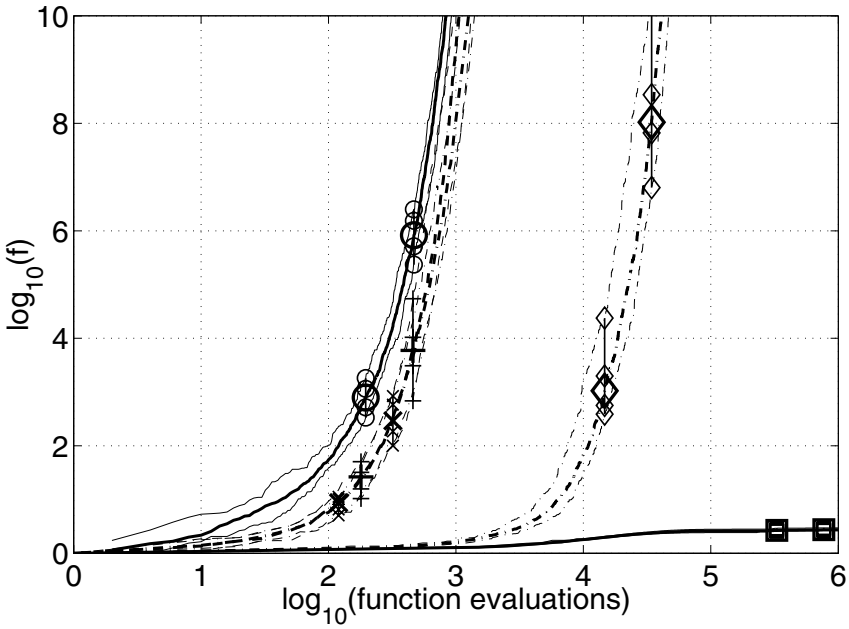

Figure 7. Diagonal plane function. $(1 + 1)$-ES ('—◯—'). Population sizes: $N_{\text{base}} = 10$ for CSA-ES ('$- \cdot - \times - \cdot -$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 1600$ for IDEA ('—□—'); $N_{\text{base}} = 100$ for MBOA ('$- \cdot -\diamond - \cdot -$').

ES. They are able to shrink fast into the subspace spanned by the coordinate axes with high scaling coefficients.

The reason for the bad performance of $(1+1)$-ES and CSA-ES is that they cannot provide different variances in different search directions. They work with a single step size only. Thus, they are not able to adapt to badly scaled functions such as f_{elli}.

By rotating the ellipsoid, the function becomes more difficult to optimize. For the $(1 + 1)$-ES, CSA-ES, and CMA-ES, the number of function evaluations to reach f_{stop} on the rotated f_{elli} remains the same as on the non-rotated f_{elli}, see Figure 10. For IDEA and MBOA, the population sizes raise by a factor of 8 and 16, respectively. The number of function evaluations increases for IDEA by the same factor. The CMA-ES is clearly the fastest strategy. MBOA performs worse by a factor of more than 1 800(!) here.

Running IDEA with $N_{\text{base}} = 1600$ on the axis-parallel ellipsoid function gives the same result as on the rotated function. The identical performance suggests that, given a sufficiently large population size, IDEA is able to precisely learn the correct dependencies. This is not the case for MBOA. In MBOA the generated search distribution does not match the objective function topology sufficiently well. The reason could either be that it is impossible

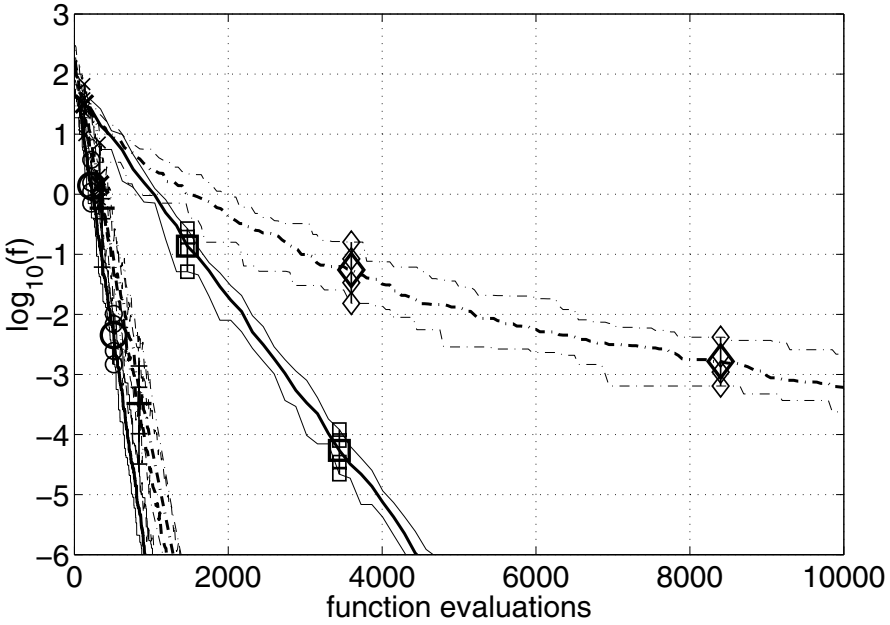

Figure 8. Sphere function. $(1+1)$-ES ('—◯—'). $N_{\text{base}} = 10$ for CSA-ES ('$-\cdot-\times-\cdot-$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 200$ for IDEA ('—□—'); $N_{\text{base}} = 100$ for MBOA ('$-\cdot-\diamond-\cdot-$').

to produce an appropriate search distribution for the rotated f_{elli} with MBOA, or that the learning procedure is inappropriate.

Why do IDEA and MBOA perform on the axis-parallel function much better than on the rotated one? With a small population size the data basis for learning the model structure is poor. As a suggestive measure, the regularization penalty enforces a structure with a low number of edges. Therefore, independent sampling is privileged and the performance in the axis-parallel case is exceptional compared with the rotated function.

4.3.4. *Cigar function*

On the axis-parallel f_{cigar}, the sequence of fastest to slowest algorithm is CMA-ES, IDEA, MBOA, CSA-ES, and $(1+1)$-ES, see Figure 11. Like f_{elli}, f_{cigar} is a badly scaled function, and the strategies with adaptation of only one step size, $(1+1)$-ES and CSA-ES, are unacceptably slow: While CMA-ES, IDEA, and MBOA reach f_{stop} in 4 000, 18 000, and 45 000 function evaluations, respectively, $(1+1)$-ES and CSA-ES need more than two million function evaluations.

In the rotated case, see Figure 12, again CMA-ES, CSA-ES, and $(1+1)$-ES have the same performance as in the non-rotated case. For IDEA and MBOA,

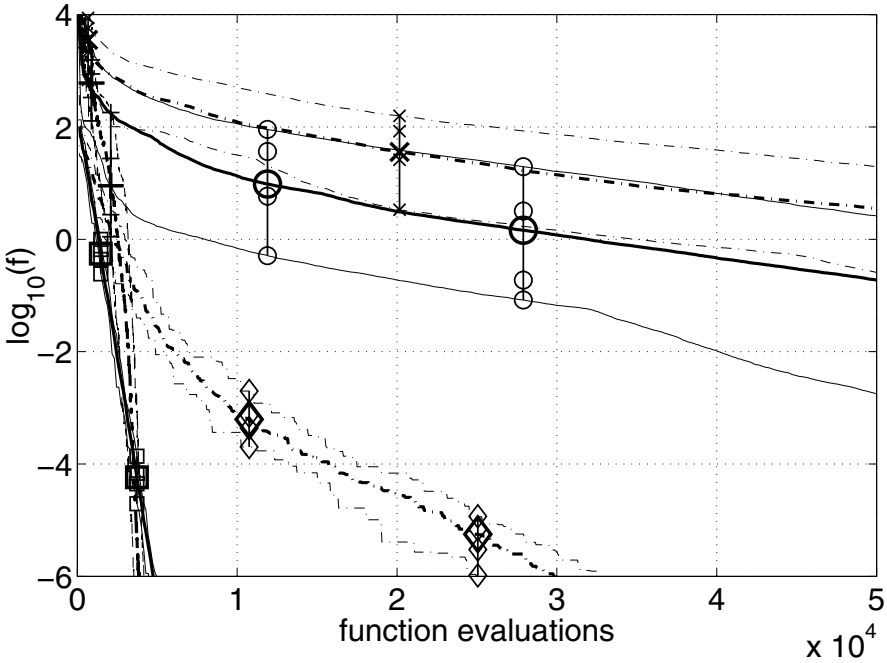

Figure 9. Ellipsoid function. $(1+1)$-ES ('—○—'). $N_{\text{base}} = 10$ for CSA-ES ('$-\cdot-\times-\cdot-$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 200$ for ID$\mathbb{E}$A ('—□—'); $N_{\text{base}} = 100$ for MBOA ('$-\cdot-\diamond-\cdot-$').

the situation is very close to that on f_{elli}: The population sizes raise by a factor of 8 and 16, respectively; for ID$\mathbb{E}$A, the increase in number of function evaluations due to the rotation corresponds to the increase in population size, and the performance with a large population size is identical in the rotated and the axis-parallel case. MBOA performs as poor as $(1 + 1)$-ES and CSA-ES.

4.3.5. *Tablet function*

The results on the axis-parallel f_{tablet}, see Figure 13, are comparable to those on the axis-parallel f_{elli} (see 4.3.3). Only the $(1 + 1)$-ES and the CSA-ES are considerably faster than on f_{elli}, although still outperformed by the other strategies.

The results on the rotated f_{tablet} are comparable to those on the rotated f_{elli}. Besides the CMA-ES, all strategies are two to three times faster than on the rotated f_{elli}. Still, the CMA-ES outperforms the second best strategy, ID$\mathbb{E}$A, by a factor of seven.

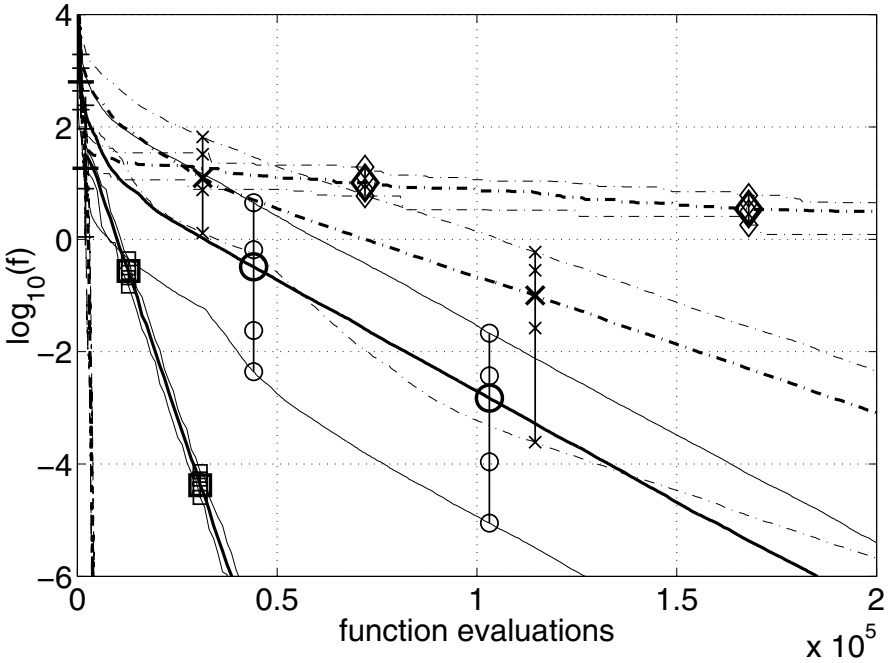

Figure 10. Rotated ellipsoid function. (1 + 1)-ES ('—○—'). $N_{\text{base}} = 10$ for CSA-ES ('−·−×−·−') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 1600$ for IDEA ('—□—'); $N_{\text{base}} = 1600$ for MBOA ('− · −◇− ·−').

4.3.6. *Rosenbrock function*

In contrast to the previous test functions, the Rosenbrock function cannot be linearly transformed into f_{sphere}. The Rosenbrock function consists of a narrow ridge. Early in the optimization, the valley floor is reached. Traveling from the valley floor to the optimum requires to change the direction continuously. We suppose that the overall direction change is almost πn radians. To efficiently traverse the search space towards the optimum, a local search mechanism, that travels along the ridge, must continuously adapt to the changing ridge direction. It is unclear whether a different, "global" search approach is possible on f_{Rosen}.

While the CMA-ES reaches f_{stop} after about 7 000 function evaluations on f_{Rosen}, the (1 + 1)-ES and the CSA-ES are 50 and 180 times slower, respectively (see Figure 15). Both MBOA and IDEA are even much slower: They do not reach values better than 0.03 and 10 after 10^6 function evaluations, respectively. These algorithms are either not able to learn a reasonable probability distribution for f_{Rosen}, or suffer from premature convergence. IDEA was tested with 1, 10, and 20 clusters and corresponding base population sizes N_{base} of up to 3 200, 16 000, and 16 000, respectively. Using this limited

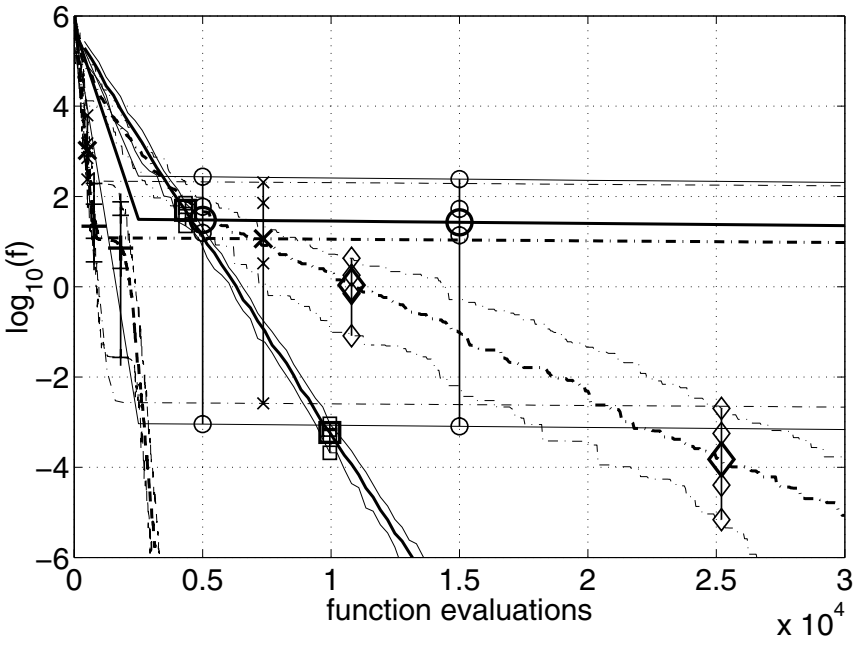

Figure 11. Cigar function. $(1 + 1)$-ES ('—◯—'). $N_{\text{base}} = 10$ for CSA-ES ('$- \cdot - \times - \cdot -$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 400$ for ID$\mathbb{E}$A ('—□—'); $N_{\text{base}} = 100$ for MBOA ('$- \cdot - \diamond - \cdot -$').

number of experiments, the clustering did not lead to a substantial improvement of the algorithm on f_{Rosen}. In contrast, Bosman and Thierens (2001) reported successful convergence within 82575 objective function evaluations for experiments with k-means clustering, where $n = 5$ and $k = 10$.

4.4. *Multimodal test functions*

4.4.1. *Rastrigin function*

The Rastrigin function is a parabolic function with a superposed cosine term of high amplitude. The number of local optima in the search region is approximately 10^n. Two successful search mechanisms are conceivable. First, the underlying global parabolic shape can guide an algorithm to the global optimum. Second, separability can be exploited, in that n one-dimensional problems with only about 10 local optima are solved.

On $f_{\text{Rastrigin}}$, see Figure 16, the $(1 + 1)$-ES is not able to locate the global optimum in any run. With sufficiently large population sizes, CMA-ES, MBOA, CSA-ES, and ID$\mathbb{E}$A can locate the global optimum reliably.

The successful location of the global optimum on $f_{\text{Rastrigin}}$ by CSA-ES and CMA-ES is a result that has not been reported before. The sufficient popula-

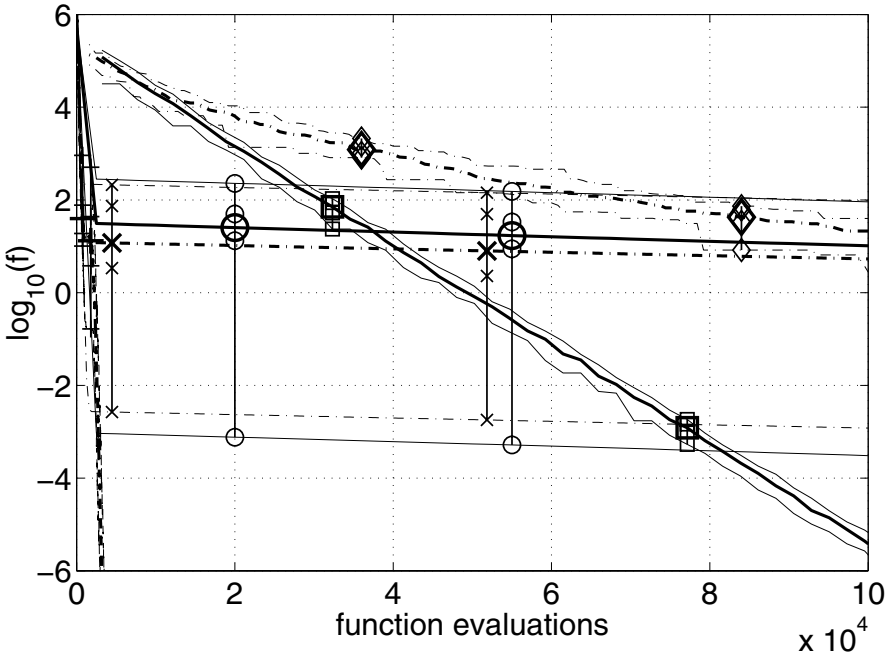

Figure 12. Rotated cigar function. (1 + 1)-ES ('—◯—'). $N_{\text{base}} = 10$ for CSA-ES ('−·−×−·−') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 3200$ for ID$\mathbb{E}$A ('—□—'); $N_{\text{base}} = 1600$ for MBOA ('− · −◇ − ·−').

tion size for CSA-ES and CMA-ES on $f_{\text{Rastrigin}}$ (where all 20 runs converge to the global optimum), is about 40 to 80 times larger than the population size $N_{\text{base}} = 10$, used for the unimodal functions in the previous sections. The difference in N_{base} between CSA-ES and CMA-ES is statistically significant. The CMA-ES needs a larger population size, because the overall variance can shrink faster than in the CSA-ES.

While $(1 + 1)$-ES, CSA-ES, and CMA-ES perform on the rotated $f_{\text{Rastrigin}}$ like on the axis-parallel $f_{\text{Rastrigin}}$, the performance of ID$\mathbb{E}$A is a factor of two slower and f_{stop} cannot be reached in all runs. Although performing well on $f_{\text{Rastrigin}}$, MBOA fails to locate the global optimum on the rotated $f_{\text{Rastrigin}}$. From our results we cannot decide whether MBOA gets trapped into a local optimum or exhibits only slow convergence. Therefore, we suppose that CMA-ES and CSA-ES exploit the global parabolic shape to locate the global optimum, while MBOA exploits the separability of $f_{\text{Rastrigin}}$. The latter supposition is supported by the previous results that show the poor performance of MBOA on rotated functions in general.

Apart from the different learning *concepts*, ID$\mathbb{E}$A and CMA-ES are similar search algorithms if the population size is large. Therefore, it is

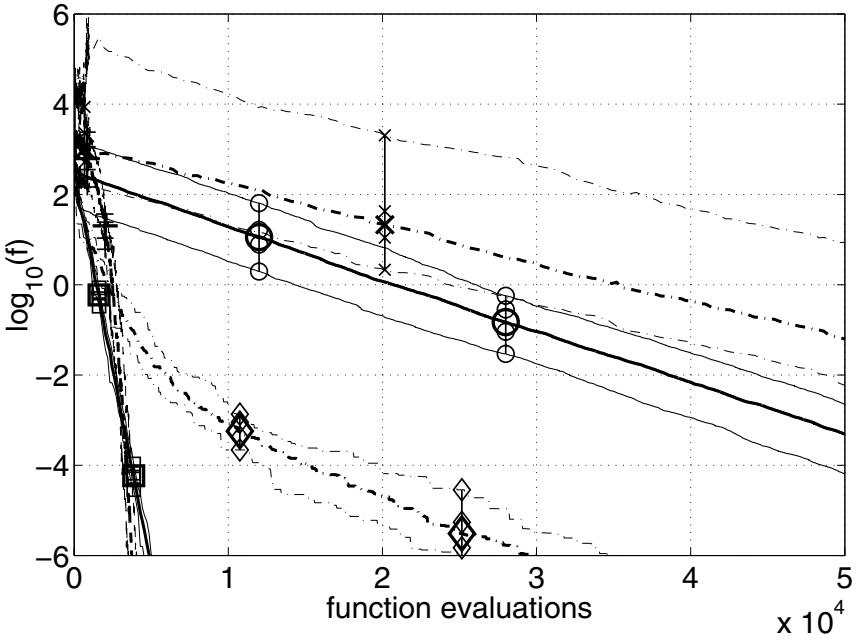

Figure 13. Tablet function. $(1+1)$-ES ('—◯—'). $N_{\text{base}} = 10$ for CSA-ES ('$-\cdot-\times-\cdot-$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 200$ for IDEA ('—□—'); $N_{\text{base}} = 100$ for MBOA ('$-\cdot-\diamond-\cdot-$').

surprising that their performance on $f_{\text{Rastrigin}}$ is so different. The difference cannot be attributed to the overall variance estimation because in the CMA-ES with large populations ($N_{\text{base}} = 800$) the estimation of the global step size is irrelevant compared with the estimation of the covariance matrix. We assume that the reason for the different performance lies in the elitist replacement strategy in IDEA: A single individual, located in a local optimum, can prevent convergence to the global optimum for a long time. This effect also explains the large variation in how long IDEA needs to reach f_{stop}.

4.4.2. *Scaled Rastrigin function*

The scaled Rastrigin function, $f_{\text{Rastrigin10}}$, differs from the Rastrigin function, $f_{\text{Rastrigin}}$, in that the coordinate axes are differently scaled. The scaling factor for "adjacent" axes is about 1.3 for $n = 10$, and the scaling factor between longest and shortest axis is ten. In both functions, local minima are located on an axes-parallel lattice. In $f_{\text{Rastrigin}}$, the distances between neighboring local optima is the same in all coordinate axes, while in $f_{\text{Rastrigin10}}$ the distances differ on the different axes.

The results on $f_{\text{Rastrigin10}}$ are very similar to those on $f_{\text{Rastrigin}}$, see Figure 18. There is one exception: The CSA-ES needs four times as many function

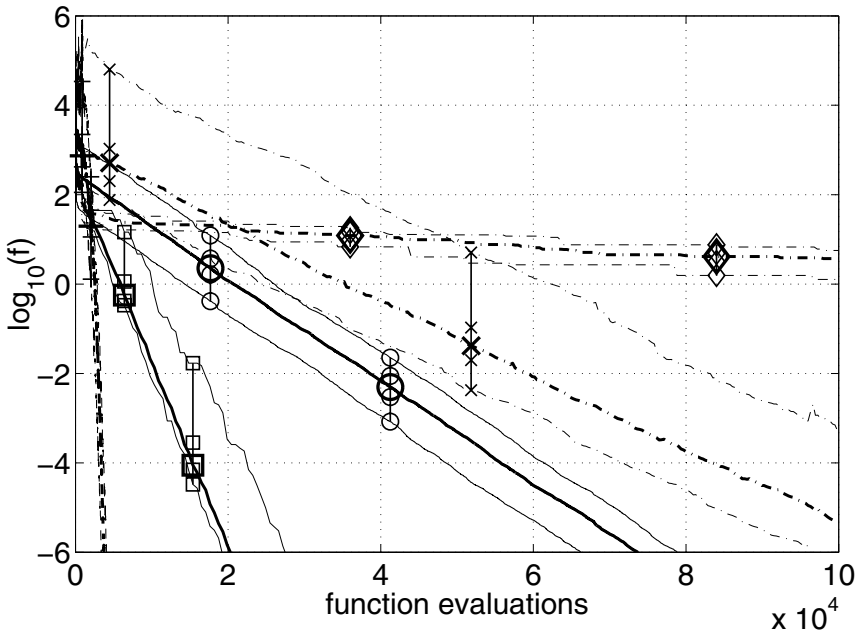

Figure 14. Rotated tablet function. (1 + 1)-ES ('—○—'). $N_{\text{base}} = 10$ for CSA-ES ('$-\cdot-\times-\cdot-$') and CMA-ES ('- - -+- - -'); $N_{\text{base}} = 1600$ for IDEA ('—□—'); $N_{\text{base}} = 800$ for MBOA ('$-\cdot-\diamond-\cdot-$').

evaluations on the scaled version. This is not surprising, because the CSA-ES cannot adapt to the scaling of the function. More surprisingly, the global optimum can still be located although the scaling cannot be learnt.

The effect of rotating $f_{\text{Rastrigin10}}$ is similar to the effect of rotating $f_{\text{Rastrigin}}$. IDEA gets slightly worse and MBOA fails to locate the global optimum, see Figure 19.

4.5. *Summary of experimental results*

The main findings of our experiments are summarized as follows:

- The (1 + 1)-ES is about 50% faster than the CSA-ES on unimodal functions. Because both strategies work with a single step size only, their performance is poor on badly-scaled functions. Rotations do not affect their behavior. On multimodal functions, the (1 + 1)-ES gets stuck in a local optimum typically. Although not designed for this purpose, the CSA-ES can be applied with large population sizes. Then, the global optimum can often be located and the CSA-ES clearly outperforms the (1 + 1)-ES on multimodal functions. Nevertheless, the rate of convergence is slow.

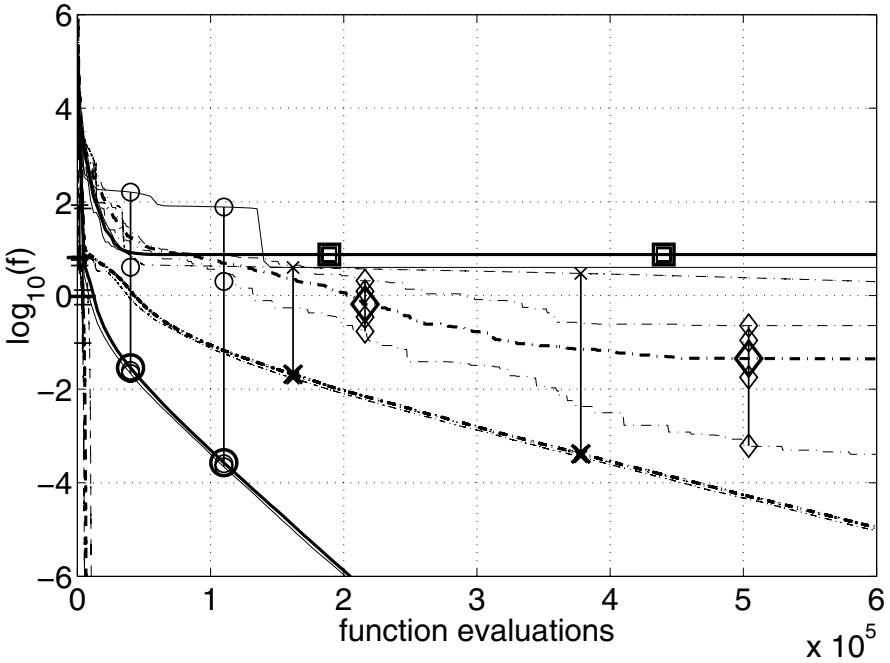

Figure 15. Rosenbrock function. (1 + 1)-ES ('—◯—'). $N_{\text{base}} = 50$ for CSA-ES ('$- \cdot - \times - \cdot -$') and CMA-ES ('- - -+- - -'); $k = 1$, $N_{\text{base}} = 3200$ for IDEA ('—□—'); $N_{\text{base}} = 1600$ for MBOA ('$- \cdot -\diamond - \cdot -$').

In conclusion, $(1+1)$-ES and CSA-ES work well only on well-scaled unimodal functions.

– The CMA-ES performs equally well on rotated and non-rotated functions. For unimodal functions, small population sizes like $N_{\text{base}} = 10$ are sufficient, and only on f_{sphere} the CMA-ES is slightly outperformed by the $(1 + 1)$-ES. On unimodal, non-separable functions the CMA-ES is by far superior to all compared algorithms.

  For multimodal functions, the CMA-ES performs surprisingly well, when sufficiently large population sizes are used. Then, the CMA-ES outperforms all other tested algorithms on the multimodal test functions.

– Compared to the ESs, IDEA needs larger population sizes even on unimodal functions. It is successful in adapting to convex-quadratic topologies, but can fail in more complex situations like the Rosenbrock function. On separable (axis-parallel) functions the sufficient population size is much smaller than on the rotated ones, because on separable functions the correct model structure is an empty structure. Then the performance is comparable to that of the CMA-ES on f_{elli} and f_{tablet}.

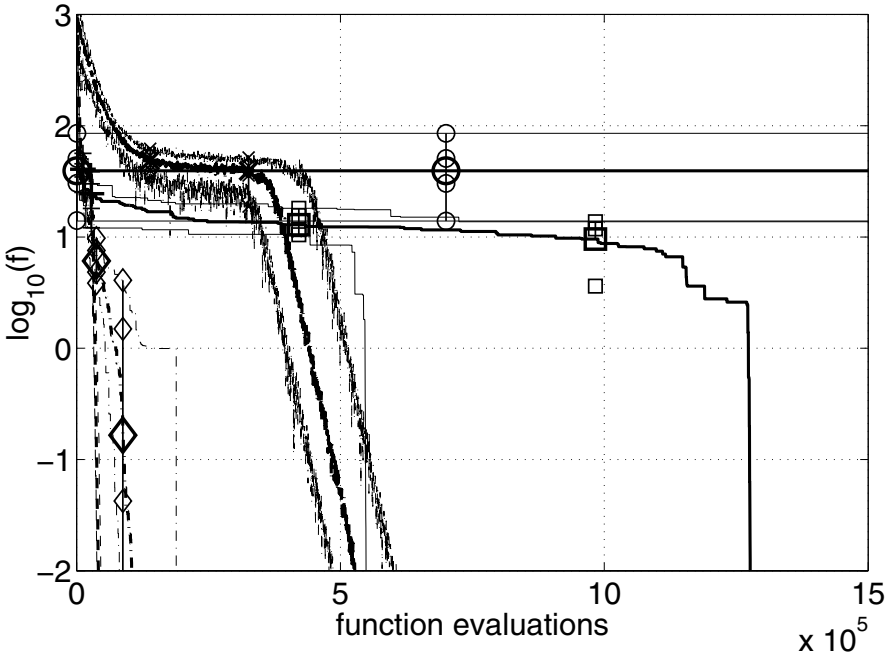

Figure 16. Rastrigin function. (1+1)-ES ('—◯—'). $N_{\text{base}} = 400$ for CSA-ES ('−·−×−·−') and $N_{\text{base}} = 800$ for CMA-ES ('- - -+- - -'); $k = 1$, $N_{\text{base}} = 400$ for IDEA ('—□—'); $N_{\text{base}} = 200$ for MBOA ('− · −◇ − ·−').

In IDEA, even on a linear function, the overall distribution variance shrinks to zero continuously. Variance shrinking in a linear test case must be regarded as a major shortcoming (Beyer and Deb, 2001). It implies that the algorithm is highly vulnerable to premature convergence.

On multimodal functions, IDEA is able to locate the global optimum, even on the non-separable functions. Probably due to the elitist replacement strategy, the convergence is slow and the performance is comparable to that of the CSA-ES.

– On unimodal separable functions, MBOA can cope with different scalings, but the convergence rate is slow. The slow convergence can be attributed to restricted tournament replacement, which on the other hand prevents premature convergence. Despite the slow convergence, MBOA performs quite well on multimodal separable functions.

  On non-separable functions, the performance of MBOA is poor. MBOA is not able to generate a reasonable search distribution in any of the non-separable test functions in our study.

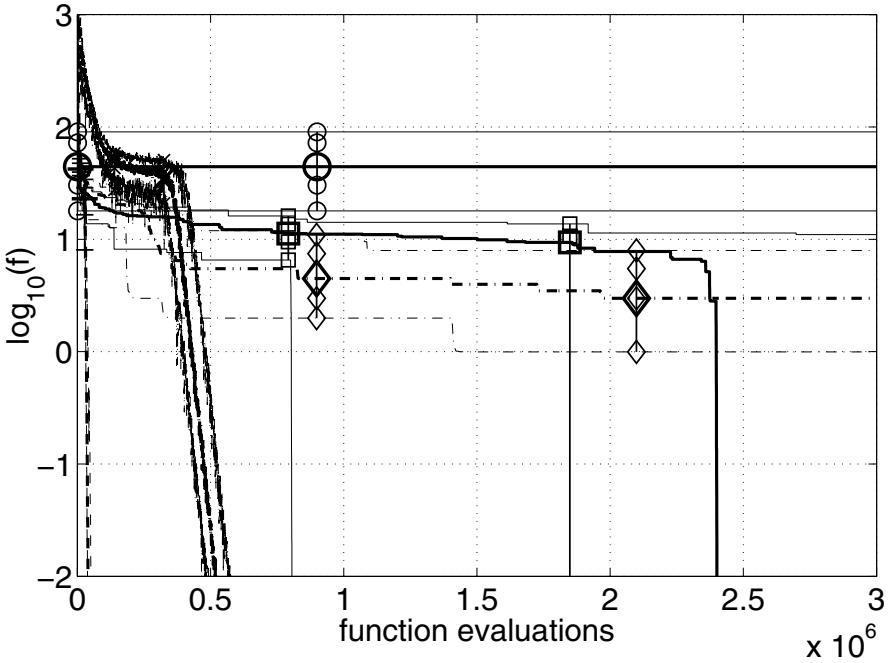

Figure 17. Rotated Rastrigin function. (1 + 1)-ES ('—◯—'). $N_{\text{base}} = 400$ for CSA-ES ('$- \cdot - \times - \cdot -$') and $N_{\text{base}} = 800$ for CMA-ES ('- - -+- - -'); $k = 1$, $N_{\text{base}} = 400$ for ID$\mathbb{E}$A ('—□—'); $N_{\text{base}} = 400$ for MBOA ('$- \cdot -\diamond - \cdot -$').

5. Summary, conclusions, and outlook

We have investigated a class of evolutionary algorithms that estimate a probability distribution from a population of individuals to sample new individuals. In particular, we studied five algorithms: (1 + 1)-ES, CSA-ES, CMA-ES, ID$\mathbb{E}$A, and MBOA by comparing their structural characteristics and by assessing their performance experimentally.

Based on their structural characteristics the algorithms can be divided into three groups. First, (1 + 1)-ES and CSA-ES sample a spherical Gaussian distribution with one degree of freedom. Second, CMA-ES and ID$\mathbb{E}$A sample an arbitrary Gaussian distribution. ID$\mathbb{E}$A additionally provides the possibility to sample a mixture of Gaussians by clustering the population. However, in our experiments mixtures of Gaussians did not show any advantage compared with the single Gaussian approach. Third, MBOA samples Gaussian kernel distributions in each variable, such that the sampling may or may not depend on the value of the previously sampled variables.

On the test functions used, the performance difference between best and worst algorithm was typically two to three orders of magnitude. Reasons

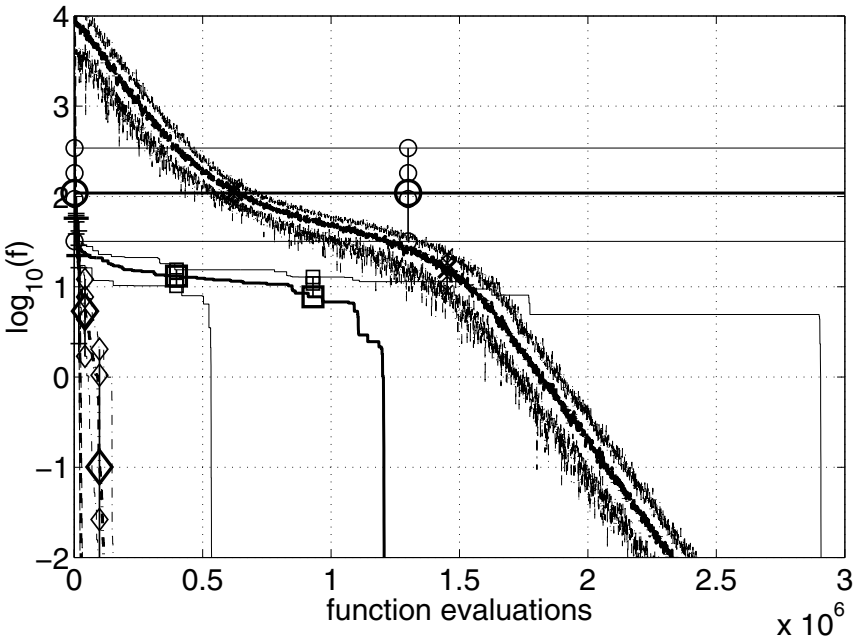

Figure 18. Scaled Rastrigin function. (1 + 1)-ES ('—◯—'). $N_{\text{base}} = 800$ for CSA-ES ('$- \cdot - \times - \cdot -$') and $N_{\text{base}} = 400$ for CMA-ES ('- - -+- - -'); $k = 1, N_{\text{base}} = 400$ for ID$\mathbb{E}$A ('—□—'); $N_{\text{base}} = 200$ for MBOA ('$- \cdot -\diamond - \cdot -$').

for performance degradation are multimodality and non-separability of the objective function. We give a performance summary, listing algorithms that perform within one order of magnitude of the best algorithm, ordered from best to worst. On well-conditioned unimodal functions the order is: (1 + 1)-ES, CMA-ES and CSA-ES; on badly scaled separable unimodal functions the order is: CMA-ES and ID$\mathbb{E}$A; on separable multimodal functions: CMA-ES and MBOA; on non-separable functions: CMA-ES. Note that different population sizes are chosen for the different test functions.

Only the ESs can be used effectively with small population sizes. This is not too surprising. First, only the ESs use additional time-varying parameters. These parameters supplement the information prevalent in the population. Second, only the ESs use explicit control concepts for the overall distribution variance. The advantage of these concepts can be seen in small populations – in particular on unimodal functions, where small population sizes are often advantageous. An efficient estimation of overall distribution variance is an open issue in ID$\mathbb{E}$A and MBOA. In ID$\mathbb{E}$A, the variance is prone to decreasing too fast, leading to premature convergence. In MBOA, a sufficiently fast change of the variance is not possible, leading to slow convergence.

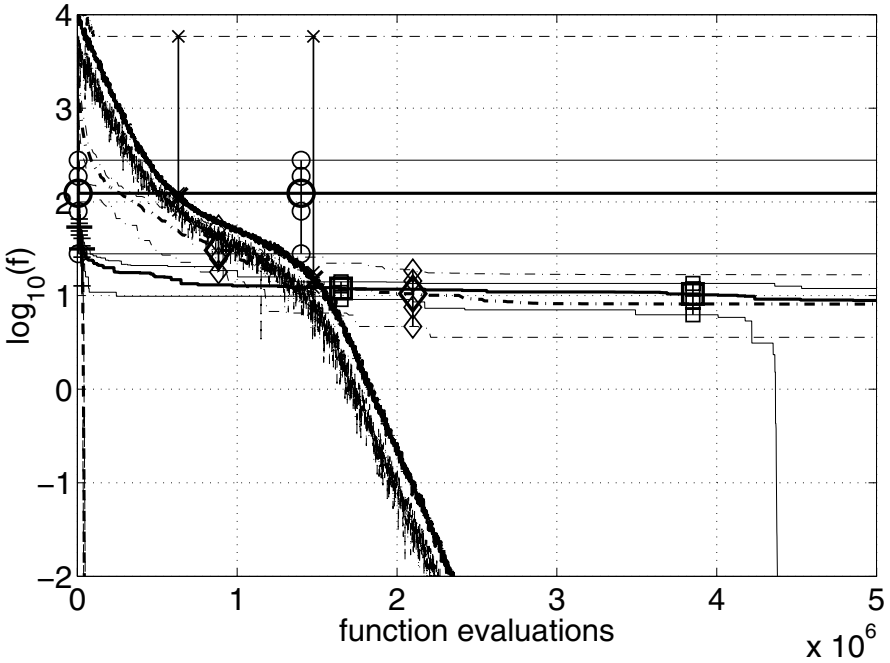

Figure 19. Rotated scaled Rastrigin function. (1+1)-ES ('—◯—'). $N_{\text{base}} = 800$ for CSA-ES ('$-\cdot-\times-\cdot-$') and $N_{\text{base}} = 800$ for CMA-ES ('- - -+- - -'); $k = 1$, $N_{\text{base}} = 1600$ for IDEA ('—□—'); $N_{\text{base}} = 3200$ for MBOA ('$-\cdot-\diamond-\cdot-$').

On multimodal functions, only CMA-ES and MBOA show competitive performance. Interestingly, these algorithms have different search strategies to achieve their global search capability. The CMA-ES relies on a topology which is characterized by a large and symmetric overall basin that leads to the global optimum. Multimodality is due to local modulations of the global structure. In contrast, MBOA relies on an objective function that is separable with respect to the given coordinate system. We note that the independent subspaces, into which the function can be separated, can have dimensions larger than one. In principle, even a limited number of dependencies between these subspaces can be covered. However, we have no empirical results supporting the advantage from this possibility in the continuous domain. While in each subspace, the optimum needs to have a reasonably large attractor basin, the overall basin volume can be very small. Therefore, non-separable functions are hard to solve for MBOA, as was demonstrated by our experiments. For the CMA-ES, functions are hard to solve where the attractor volume of the global optimum is small, and an overall topology pointing to the global optimum is missing.

Our experimental investigation was conducted for problem dimension $n = 10$. The comparison of the scaling of the algorithms is part of ongoing investigations. This comparison appears to be interesting in particular in the multimodal case, where the suite of test functions should be extended as well. For multimodal functions, the different search strategies might reveal significantly different scaling behaviors.

References

Baluja S and Caruana R (1995) Removing the Genetics from the Standard Genetic Algorithm. Technical report, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. CMU-CS-95-141

Beyer H-G and Deb K (2001) On self-adaptive features in real-parameter evolutionary algorithms. IEEE Transactions on Evolutionary Computation 5(3): 250–270

Beyer H-G and Schwefel H-P (2002) Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52

Bosman PAN and Thierens D (2000a) Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo JJ and Schwefel H-P (eds) Parallel Problem Solving from Nature – PPSN VI, pp. 767–776. Springer, Berlin

Bosman PAN and Thierens D (2000b) Mixed IDEAs. Technical report, Utrech University. UU-CS-2000-45

Bosman PAN and Thierens D (2001) Advancing continuous IDEAs with mixture distributions and factorization selection metrics. In: Optimization by Building and Using Probabilistic Models (OBUPM) 2001, pp. 208–212. San Francisco, California, USA

Hansen N, Müller SD and Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1): 1–18

Hansen N and Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of the 1996 IEEE Conference on Evolutionary Computation (ICEC '96), pp. 312–317

Hansen N and Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2): 159–195

Larrañaga P (2002) A review on estimation of distribution algorithms. In: Larrañaga P and Lozano JA (eds) Estimation of Distribution Algorithms, pp. 80–90. Kluwer Academic Publishers, Dordrecht

Mühlenbein H and Paass G (1996) From recombination of genes to the estimation of distributions: I. binary parameters. Lecture Notes in Computer Science 1141: 178–187

Ocenasek J and Schwarz J (2002) Estimation of distribution algorithm for mixed continuous – discrete optimization problems. In: 2nd Euro-International Symposium on Computational Intelligance, pp. 227–232. IOS Press, Kosice, Slowakia

Ostermeier A, Gawelczyk A and Hansen N (1994) Step-size adaptation based on non-local use of selection information. In: Davidor Y, Schwefel H-P and Männer R (eds) Parallel Problem Solving from Nature – PPSN IV, Proceedings, Jerusalem, pp. 189–198. Springer, Berlin

Parzen E (1962) On estimation of probability density function and mode. Annual Mathematical Statistics 33: 1065–1076

Pelikan M and Goldberg DE (2001) Escaping hierarchical traps with competent genetic algorithms. IlliGAL Report No. 2001003, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL

Pelikan M, Goldberg DE and Cantú-Paz E (1999) BOA: The Bayesian optimization algorithm. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M and Smith RE (eds) Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Vol. I. Orlando, FL, pp. 525–532. Morgan Kaufmann Publishers, San Francisco, CA

Pelikan M, Goldberg DE and Sastry K (2000) Bayesian optimization algorithm, decision graphs, and Occam's razor. IlliGAL Report No. 2000020, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL

Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart

Schwefel H-P (1995) Evolution and Optimum Seeking. John Wiley & Sons, New York