Information Sciences 182 (2012) 199–219

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

A Differential Covariance Matrix Adaptation Evolutionary Algorithm for real parameter optimization

Saurav Ghosh [a], Swagatam Das [b,*], Subhrajit Roy [a], S.K. Minhazul Islam [a], P.N. Suganthan [c]

[a] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India
[b] Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700 108, India
[c] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 22 January 2011
Received in revised form 6 August 2011
Accepted 13 August 2011
Available online 25 August 2011

Keywords:
Differential Evolution
Covariance Matrix Adaptation Evolutionary Strategy
Numerical optimization
Explorative power
Population variance

ABSTRACT

Hybridization in context to Evolutionary Computation (EC) aims at combining the operators and methodologies from different EC paradigms to form a single algorithm that may enjoy a statistically superior performance on a wide variety of optimization problems. In this article we propose an efficient hybrid evolutionary algorithm that embeds the difference vector-based mutation scheme, the crossover and the selection strategy of Differential Evolution (DE) into another recently developed global optimization algorithm known as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). CMA-ES is a stochastic method for real parameter (continuous domain) optimization of non-linear, non-convex functions. The algorithm includes adaptation of covariance matrix which is basically an alternative method of traditional Quasi-Newton method for optimization based on gradient method. The hybrid algorithm, referred by us as Differential Covariance Matrix Adaptation Evolutionary Algorithm (DCMA-EA), turns out to possess a better blending of the explorative and exploitative behaviors as compared to the original DE and original CMA-ES, through empirical simulations. Though CMA-ES has emerged itself as a very efficient global optimizer, its performance deteriorates when it comes to dealing with complicated fitness landscapes, especially landscapes associated with noisy, hybrid composition functions and many real world optimization problems. In order to improve the overall performance of CMA-ES, the mutation, crossover and selection operators of DE have been incorporated into CMA-ES to synthesize the hybrid algorithm DCMA-EA. We compare DCMA-EA with original DE and CMA-EA, two best known DE-variants: SaDE and JADE, and two state-of-the-art real optimizers: IPOP-CMA-ES (Restart Covariance Matrix Adaptation Evolution Strategy with increasing population size) and DMS-PSO (Dynamic Multi Swarm Particle Swarm Optimization) over a test-suite of 20 shifted, rotated, and compositional benchmark functions and also two engineering optimization problems. Our comparative study indicates that although the hybridization scheme does not impose any serious burden on DCMA-EA in terms of number of Function Evaluations (FEs), DCMA-EA still enjoys a statistically superior performance over most of the tested benchmarks and especially over the multi-modal, rotated, and compositional ones in comparison to the other algorithms considered here.

* Corresponding author.
E-mail addresses: saurav_online@yahoo.in (S. Ghosh), swagatam.das@isical.ac.in (S. Das), roy.subhrajit20@gmail.com (S. Roy), skminha.isl@gmail.com (S.K. Minhazul Islam), epnsugan@ntu.edu.sg (P.N. Suganthan).

0020-0255/$ - see front matter © 2011 Published by Elsevier Inc.
doi:10.1016/j.ins.2011.08.014

1. Introduction

Scientists and engineers from all disciplines often have to deal with the classical problem of global optimization where the main target is to determine a set of model parameters or state-variables that provide the globally minimum or maximum value of a predefined cost or objective function, or a set of optimal trade-off values in the case of two or more conflicting objectives. In this article we consider bound-constrained single-objective optimization problems that involve D decision variables represented in a vector like $\vec{X} = [x_1, x_2, x_3, \ldots, x_D]^T$ and a scalar objective function (or fitness function) to judge the quality of the solution that we have achieved. The task of optimization is basically a search for such the parameter vector $\vec{X}^*$, which minimizes such an objective function $f(\vec{X})(f : \Omega \subseteq \Re^D \to \Re)$, i.e. $f(\vec{X}^*) < f(\vec{X})$ for all $\vec{X} \in \Omega$, where Ω is a non-empty, large but bounded set serving as the domain of the search. Since $\max\{f(\vec{X})\} = -\min\{-f(\vec{X})\}$, the restriction to minimization is without loss of generality. In general the optimization task is complicated by the existence of non-linear objective functions with multiple local minima. A local minimum $f_\ell = f(\vec{X}_\ell)$ may be defined as: $\exists\varepsilon > 0 \forall \vec{X} \in \Omega : \|\vec{X} - \vec{X}_\ell\| < \varepsilon \Rightarrow f_\ell \leqslant f(\vec{X})$, where $\|\cdot\|$ indicates any p-norm distance measure. Locating the global optimal solutions becomes very challenging for single-objective problems arising in many practical situations, due to the presence of high dimensionality, strong epistasis, ill-conditioning, and multimodality of the objective function.

Hybridization, in context to evolutionary computing, primarily refers to the process of combining the best features of two or more algorithms together, to form a new EA that is expected to outperform its ancestors over application-specific or general benchmark problems. A common template for hybridization is provided by the Memetic Algorithms (MAs), which combine the respective advantages of global search and local search [19,35]. There exist also a plethora of examples where hybridization of two or more global optimization algorithms have resulted into improved performance with greater accuracy and robustness, e.g. see [2,11,31,42,45]. Hybridization of EAs is getting popular due to their capabilities in handling several real world problems involving complexity, noisy environment, imprecision, uncertainty, and vagueness. In this paper we propose a simple yet very powerful hybrid EA that synergistically combines the features of two global optimizers: Differential Evolution (DE) [6,10,43,46] and Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [14]. DE has emerged as a very competitive optimizer for continuous search spaces, exhibiting remarkable performances in several competitions held under the Institute for Electrical and Electronic Engineers (IEEE) Congress on Evolutionary Computation (CEC) [25,47,48,57]. In recent times, researchers are focussed on improving the performance of DE over complicated landscapes by introducing innovative modifications in mutation, crossover and parameter adaptation strategies [29,30,32]. In DE community, the individual trial solutions (which constitute a population) are called *parameter vectors* or *genomes*. Unlike traditional EAs, DE perturbs the current-generation vectors with the scaled differences of two randomly selected and mutually distinct population vectors. Therefore, in contrast to the classical evolutionary computing paradigms like Evolutionary Programming (EP) or Evolution Strategies (ESs), no separate probability distribution has to be used for generating the offspring. A detailed survey of the DE family of algorithms can be found in [6]. On the other hand, CMA-ES [15–18] is also a very competitive real-parameter optimizer for continuous search spaces. Compared to other evolutionary algorithms, an important property of CMA-ES is its invariance against the linear transformations in the search space. In other words it exhibits the same performances for a given objective functions $f : x \in \Re^n \mapsto f(x) \in \Re$ where $n \in N$ or for the same function where a linear transformation is applied, i.e. $f_R : x \in \Re^n \mapsto f(Rx) \in \Re$ where R denotes the linear transformation. In practice this transformation is learned by the CMA [12,13,23,24].

The main problem regarding CMA-ES is that for a fitness space if it is able to rescale the ellipsoid projection of the multivariate normal distribution into a spherical one by setting the covariance matrix of the search distribution to the inverse Hessian matrix then it is able to show successful convergence performance for that given problem. But as the landscapes get more and more complex, CMA-ES fails to convert it into a spherical projection and becomes liable to get stuck in a local optimum. The performance of CMA-ES deteriorates to a large extent when it encounters noisy, hybrid composition functions as well as several complex real-life applications. In the proposed hybridization our aim is to incorporate the difference-vector based mutation scheme of DE into CMA-ES as these difference vectors has the ability to adjust to the natural scaling of the problem overcoming the above mentioned limitations of CMA-ES. Further, in order to enhance the diversity among the population members as well as increase the convergence speed, the selection and crossover operators of DE have also been embedded.

In our proposed hybrid scheme, three most important features of DE such as the difference-vector based mutation; binomial crossover operation, and the selection procedure are incorporated into the structure of a CMA-ES algorithm in order to achieve a better performance than both of the ancestors. In what follows we shall refer to this hybrid algorithm as Differential Covariance Matrix Adaptation Evolutionary Algorithm (DCMA-EA). In our proposed hybrid scheme, the newly generated vector in the next population is created by taking a controlled share of the current generation mean and its target. It also takes the help of a scaled difference from two vectors of the current generation. Finally it also adds the controlled step-size for the current generation guided by the covariance matrix adaptation. We also have introduced the selection and crossover strategy of the DE to have its own advantage in an evolutionary algorithm. The proposed scheme takes the advantage of the CMA-ES of overcoming badly scaled or high non-separable functions into premature convergence and also utilizes the DE with the introduction of crossover which ensures component injection and selection [7,53,54,56] ensuring of absence of any inferior vectors in the next generations. Introduction of the DE-type mutation, crossover and selection do not impose any serious statistical and computational burden on the original structure of CMA-ES as these operations do not demand additional number of Function Evaluations (FEs).

In order to experimentally validate the effectiveness of DCMA-EA, we choose a test-suite comprising of 20 numerical benchmarks with varied range of complexity and an engineering optimization problem involving the spread-spectrum radar poly-phase code design and Frequency Modulator Synthesis problem. To illustrate the effectiveness of the incorporation of DE-type mutation operators, binomial crossover scheme and the selection process we compare its performance with those of the original DE and the original CMA-ES. To judge how much improvement we achieve over the recently developed and best-known DE-variants, we compare DCMA-EA with the Self-Adaptive DE (SaDE) [44] and an adaptive DE with DE/current-to-*p*best/1 mutation scheme and optional external archive (JADE) [56]. Finally we also compare DCMA-EA with two very powerful and competitive real parameter optimizers of current interest: the IPOP-CMA-ES (a Restart Covariance Matrix Adaptation Evolution Strategy with increasing population size) [1] and the DMS-PSO (Dynamic Multi Swarm Particle Swarm Optimizer) [26]. Performances of all these algorithms are tested on the same benchmark set and under homogeneous experimental conditions so that an unbiased comparison may be obtained. Our study indicates that DCMA-EA is superior to or comparable with some of the most competitive real-parameter optimizers in a statistically meaningful way.

2. Classical CMA-ES

The CMA-ES is a stochastic method for real parameter (continuous domain) optimization of non-linear, non-convex functions. The algorithm includes adaptation of covariance matrix which is basically an alternative method of traditional Quasi-Newton method for optimization based on gradients. Here are some basic tools for realizing the CMA-ES algorithm.

2.1. Eigen decomposition of a positive definite matrix

A symmetric positive definite matrix, $C \in R^{n\times n}$, is defined such a way that for all $x \in R^n \backslash \{0\}$ holds $x^T C x > 0$. The matrix C has an orthogonal basis of eigenvectors, $B = [b_1, b_2, \ldots, b_n]$, with corresponding eigenvalues, $d_1^2, \ldots, d_n^2 > 0$. That means for each b_i holds

$$Cb_i = d_1^2 b_i. \tag{1}$$

It has been assumed here that the orthogonal eigenvectors are of unit length, $b_i^T b_j = \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$, and $B^T B = I$. Now, for any $v \in R^n$ we can find coefficients α_i, such that $v = \sum_i \alpha_i b_i$, and then we have $Cv = \sum_i d_i^2 v \alpha_i b_i$. So, eigen decomposition of C obeys

$$C = BD^2 B^T, \tag{2}$$

Here, B is an orthogonal matrix; $B^T B = BB^T = I$. Columns of B forms an orthonormal basis of eigenvectors. $D^2 = DD = diag(d_1, \ldots, d_n)^2 = diag\left(d_1^2, \ldots, d_n^2\right)$ is a diagonal matrix with eigenvalues of C as diagonal elements. $D = diag(d_1, \ldots, d_n)$ is a diagonal matrix with square root of eigenvalues of C as the diagonal elements.

$$C^{-1} = (BD^2 B^T)^{-1} = B^{T^{-1}} D^{-2} B^{-1} = BD^{-2} B^T = B diag\left(\frac{1}{d_1^2}, \ldots, \frac{1}{d_n^2}\right) B^T, \tag{3}$$

From Eq. (2) we naturally define

$$C^{\frac{1}{2}} = BDB^T. \tag{4}$$

And therefore

$$C^{-\frac{1}{2}} = BD^{-1} B^T = B\, diag\left(\frac{1}{d_1}, \ldots, \frac{1}{d_n}\right) B^T. \tag{5}$$

2.2. Introduction to multivariate normal distribution

A multivariate normal distribution $N(m,C)$, has a uni-modal, bell-shaped density, where the modal value corresponds to the distribution mean m.The distribution $N(m,C)$ is uniquely determined by its mean $m \in R^n$ and it is symmetric and positive definite covariance matrix $C \in R^{n\times n}$. The normal distribution $N(m,C)$ can be written in different ways

$$N(m,C) \sim m + N(m,C) \sim m + C^{\frac{1}{2}} N(0,I) \sim m + BD \underbrace{B^T N(0,I)}_{\sim N(0,I)} \sim m + BD \underbrace{N(0,I)}_{\sim N(0,D^2)} \tag{6}$$

where "$\sim$" denotes equality in distribution. $N(0,I)$ produces a spherical (isotropic) distribution.

2.3. Hessian and covariance matrix

For the discussion of Hessian matrix at first a convex-quadratic objective function $f_H : x \mapsto \frac{1}{2}x^T Hx$ is considered where the Hessian matrix $H \in R^{n\times n}$ is a positive definite matrix. For a search distribution $N(m, C)$, there is a close relation between H and C: Setting $C = H^{-1}$ on f_H is equivalent to optimize the isotropic function $f_{sphere}(x) = \frac{1}{2}x^T x = \frac{1}{2}\sum_i x_i^2$ (where $H = I$) with $C = I$. So, on convex quadratic objective functions setting the covariance matrix of a search distribution to the inverse Hessian matrix is equivalent to rescaling the ellipsoid function into a spherical one. Furthermore, choosing a covariance matrix or choosing a respective affine linear transformation of the search space (i.e. of x) is equivalent because for any full rank $n \times n$ matrix A we find a positive definite Hessian matrix such that $\frac{1}{2}(Ax)^T Ax = \frac{1}{2}x^T A^T Ax = \frac{1}{2}x^T Hx$.

2.4. Sampling

The new population vectors in the subsequent generations are generated in the CMA evolution strategy by means of sampling which takes the help of the following equation:

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)} N\left(0, C^{(g)}\right). \tag{7}$$

Here $\sim$ denotes the same distribution on the left and right side. $N(0, C^{(g)})$ is a multivariate normal distribution with zero mean with covariance matrix $C^{(g)}$. $x_k^{(g+1)} \in R^n$ is kth individual from generation $g + 1$. $m^{(g)} \in R^n$ is mean value of the search distribution at generation g. $\sigma^{(g)} \in R_+$ is overall standard deviation (step-size) at generation g. $\lambda \geqslant 2$ is the population size (sample size).

2.5. Recombination

The new mean $m^{(g+1)}$ of the all current population vectors is the weighted average of μ selected vectors from the samples $x_1^{(g+1)}, \ldots, x_\lambda^{(g+1)}$:

$$m^{(g+1)} = \sum_{i=0}^{\mu} w_i x_{i:\lambda}^{(g+1)}. \tag{8}$$

Here

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geqslant w_2 \geqslant \cdots \geqslant w_\mu > 0. \tag{9}$$

Here, $\mu \leqslant \lambda$ (parent population size) is the number of selected points from the whole population. $w_{i=1,\,2,\ldots,\mu} \in R_+$ are positive weight coefficients for recombination. $x_{i:\lambda}^{(g+1)}$ is the ith best individual out of $x_1^{(g+1)}, \ldots, x_\lambda^{(g+1)}$ from Eq. (7). The index $i{:}\lambda$ denotes the index of the ith ranked individuals and $f(x_{1:\lambda}^{(g+1)}) \leqslant f(x_{2:\lambda}^{(g+1)}) \leqslant f(x_{3:\lambda}^{(g+1)}) \leqslant \cdots \leqslant f(x_{\lambda:\lambda}^{(g+1)})$, where f is the objective function to be minimized. Here, a new term related to μ is introduced whose measure

$$\mu_{eff} = \left(\frac{\|w\|_1}{\|w\|_2}\right)^2 = \frac{\|w\|_1^2}{\|w\|_2^2} = \frac{1}{\|w\|_2^2} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1} \tag{10}$$

is termed as variance effective selection mass.

2.6. Covariance matrix adaptation

In this section the adaptation of covariance matrix has been derived which is the essence of the CMA-ES algorithm. At first, instant estimation of the covariance matrix is carried out from a single population of one generation. The estimation is unreliable for very small population and an adaptation procedure named rank-μ-update is normally done. In certain limit case only a single vector can be used to update the covariance matrix at each generation which is known as rank-one-update. Finally, the two updating methods are combined together.

2.6.1. Estimating the covariance without any update

At first, the original covariance matrix $C^{(g)}$ can be estimated using the sample population from Eq. (7), $x_1^{(g+1)}, \ldots, x_\lambda^{(g+1)}$, via the empirical covariance matrix

$$C_{emp}^{(g+1)} = \frac{1}{\lambda - 1}\sum_{i=1}^{\lambda}\left(x_i^{(g+1)} - \frac{1}{\lambda}\sum_{j=1}^{\lambda} x_j^{(g+1)}\right)\left(x_i^{(g+1)} - \frac{1}{\lambda}\sum_{j=1}^{\lambda} x_j^{(g+1)}\right)^T. \tag{11}$$

The empirical covariance matrix $C_{emp}^{(g+1)}$ is an estimator of $C^{(g)}$: assuming the $x_i^{(g+1)}, i = 1, \ldots, \lambda$, to be random variables (rather than a realized sample) it is said that $E\left[C_{emp}^{(g+1)} | C^{(g)}\right] = C^{(g)}$. Considering a slight different approach to get an estimator for $C^{(g)}$

$$C_{\lambda}^{(g+1)} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left(x_i^{(g+1)} - m^{(g)}\right)\left(x_i^{(g+1)} - m^{(g)}\right)^T. \tag{12}$$

The later equation is also another unbiased estimator of $C^{(g)}$. To estimate a better co-variance matrix Eq. (12) is modified and the same weighted selection mechanism of Eq. (8) is used

$$C_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} \left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)\left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)^T. \tag{13}$$

The matrix $C_{\mu}^{(g+1)}$ is an estimator for $C^{(g)}$. Sampling from $C_{\mu}^{(g+1)}$ tends to reproduce selected, i.e. successful steps giving a justification for a better co-variance matrix.

2.6.2. Rank μ-update

To achieve fast search (opposite to more robust and more global search) the population size and μ_{eff} should be quite small ($\mu_{eff} \leqslant 1 + \ln n$). Then it is not possible to get a reliable estimator for a good co-variance matrix from Eq. (13). So as a remedy, information from previous generation is used additionally. After a sufficient number of generations the mean of the estimated co-variance matrices from all generations,

$$C^{(g+1)} = \frac{1}{g+1} \sum_{i=0}^{g} \frac{1}{\sigma^{(i)^2}} C_{\mu}^{(i+1)}, \tag{14}$$

becomes a reliable estimator for the selected steps. To make $C_{\mu}^{(g)}$ from different generations comparable, the different $\sigma^{(i)^2}$ are incorporated. In Eq. (14), all generation steps have the same weight. To assign recent generations a higher weight, exponential smoothing is introduced. Choosing $C^{(0)} = I$ to be unity matrix and a learning rate $0 \leqslant c_{\mu} \leqslant 1$, then $C^{(g+1)}$ is given by

$$C^{(g+1)} = (1 - c_{\mu})C^{(g)} + c_{\mu} \frac{1}{\sigma^{(g)^2}} C_{\mu}^{(g+1)} = (1 - c_{\mu})C^{(g)} + c_{\mu} \sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g+1)} y_{i:\lambda}^{(g+1)^T}. \tag{15}$$

Here, $c_{\mu} \leqslant 1$ is the learning rate for updating the co-variance matrix. For $c_{\mu} = 1$ no prior information is retained and $C^{(g+1)} = \frac{1}{\sigma^{(g)^2}} C_{\mu}^{(g+1)}$. For $c_{\mu} = 0$, no learning takes place and $C^{(g+1)} = C^{(0)}$. Here $c_{\mu} \approx \min(1, \mu_{eff}/n^2)$ is the reasonable choice and $y_{i:\lambda}^{(g+1)} = (x_{i:\lambda}^{(g+1)} - m^{(g)})/\sigma^{(g)}$. This co-variance matrix update is called Rank- μ-update, because the sum of outer products in Eq. (15) is of rank $\min(\mu, n)$(with probability 1). The choice of c_{μ} is crucial. Small values lead to slow learning, too large values lead to failure, because the covariance matrix degenerates. A good setting should be largely dependent on the function landscape. A first order approximation for a good choice is $c_{\mu} = \mu_{eff}/n^2$.

2.6.3. Rank-one-update

In section 2.6.1 the complete covariance matrix has been estimated from scratch, using all selected steps from a single generation. But now, the covariance matrix in the generation's sequence would be repeatedly updated using a single selected step only. Firstly, this perspective will give a justification of the adaption rule in the Eq. (15). Secondly a so-called evolution path is finally used for a rank-one update of the covariance matrix.

At first a specific method has been considered to produce n-dimensional normal distribution with zero mean. Let the vectors $y_1, \ldots, y_{g0} \in R^n$, $g0 \geqslant n$, span R^n and let $N(0,1)$ denote independent $(0,1)$-normally distributed random numbers, then

$$N(0,1)y_1 + \cdots + N(0,1)y_{g0} \sim N\left(0, \sum_{i=1}^{g0} y_i y_i^T\right) \tag{16}$$

is a normally distributed random vector with zero mean and co-variance matrix $\sum_{i=1}^{g0} y_i y_i^T$. The singular distribution $N(0,1)y_i \sim N(0, y_i y_i^T)$ generates the vector y_i with maximum likelihood considering all normal distributions with zero mean. Let the sum in Eq. (15) consist of a single summand only (e.g. $\mu = 1$), and let $y_{g+1} = \frac{x_{1:\lambda}^{(g+1)} - m(g)}{\sigma^{(g)}}$. Then the rank-one-update for the covariance matrix is given by

$$C^{(g+1)} = (1 - c_1)C^{(g)} + c_1 y_{g+1} y_{g+1}^T. \tag{17}$$

2.6.4. Cumulation: utilizing the evolution path

To update the covariance matrix in Eqs. (14) and (17) the selected step, $y_{i:\lambda} = \frac{x_{1:\lambda}^{(g+1)} - m(g)}{\sigma^{(g)}}$ is used. An evolution path can be expressed by a sum of consecutive steps. This summation is referred to as cumulation. To construct an evolution path the step size σ is constructed by the sum

$$\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}}=\frac{m^{(g)}-m^{(g-1)}}{\sigma^{(g-1)}}=\frac{m^{(g-1)}-m^{(g-2)}}{\sigma^{(g-2)}}. \tag{18}$$

In practice, to construct the evolution path $p_c \in R^n$ it is started with $p_c^{(0)}=0$.

$$p_c^{(g+1)}=(1-c_c)p_c^{(g)}+\sqrt{c_c(2-c_c)\mu_{eff}}\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}}. \tag{19}$$

Here, $p_c \in R^n$ is the evolution path at generation g. $c_c \leqslant 1$. The factor $\sqrt{c_c(2-c_c)\mu_{eff}}$ is normalization constant for p_c. For $c_c = 1$ and $\mu_{eff} = 1$, the factor reduces to one and $p_c^{(g+1)}=\frac{x_{1,\lambda}^{(g+1)}-m(g)}{\sigma^{(g)}}$. So the rank-one-update of the covariance matrix $C^{(g)}$ via the evolution path $p_c^{(g+1)}$ is given by

$$C^{(g+1)}=(1-c_1)C^{(g)}+c_1 p_c^{(g+1)}p_c^{(g+1)^T}. \tag{20}$$

As a last step Eqs. (14) and (17) has been combined.

2.6.5. Combining rank-μ-update and cumulation

The final covariance matrix adaptation update of the covariance matrix combines Eqs. (14) and (17):

$$C^{(g+1)}=(1-c_1-c_\mu)C^{(g)}+c_1\underbrace{p_c^{(g+1)}p_c^{(g+1)^T}}_{\text{rank-one update}}+c_\mu\underbrace{\sum_{i=1}^{\mu}w_i y_{i:\lambda}^{(g+1)}\left(y_{i:\lambda}^{(g+1)}\right)^T}_{\text{rank-}\mu\text{ update}} \tag{21}$$

Here, $c_1 \approx 2/n^2$. $c_\mu \approx \min(\mu_{eff}/n^2, 1-c_1)$. $y_{i:\lambda}^{(g+1)}=\left(x_{i:\lambda}^{(g+1)}-m^{(g)}\right)\Big/\sigma^{(g)}$.

Eq. (21) reduces to 14 for $c_1 = 0$ and to 26 for $c_\mu = 0$. The equation combines the advantages of 14 and 26. The information within the population of one generation is used efficiently by the rank-μ-update. On the other hand information of correlations between generations is exploited by using the evolution path for the rank-one update. The former is important in large populations while the latter is in particular important in small populations.

2.7. Step-size control

The covariance matrix adaptation, introduced in the last section, does not explicitly control the overall scale of the distribution, the step-size. The covariance matrix adaptation increases the scale only in one direction for each selected step, and it decreases the scale only implicitly by fading out the old information via the factor $1-c_1-c_\mu$. To control the step size $\sigma^{(g)}$ the evolution path is utilized. To construct the evolution path, p_c, from Eq. (22) depends on its direction. Initialized with $p_\sigma^{(0)}=0$ is given by

$$p_\sigma^{(g+1)}=(1-c_\sigma)p_\sigma^{(g)}+\sqrt{c_\sigma(2-c_\sigma)\mu_{eff}}C^{(g)^{-\frac{1}{2}}}\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}}. \tag{22}$$

Here $p_\sigma^{(g)}$ is the conjugate evolution path at generation g. $\sqrt{c_\sigma(2-c_\sigma)\mu_{eff}}$ is a normalization constant. $C^{(g)^{-\frac{1}{2}}}=B^{(g)}D^{(g)^{-1}}B^{(g)^T}$, where $C^{(g)}=B^{(g)}(D^{(g)})^2B^{(g)^T}$ is an eigen decomposition of $C^{(g)}$ where $B^{(g)}$ is an orthonormal basis of eigenvectors, and the diagonal elements of the diagonal matrix $D^{(g)}$ are square roots of the corresponding positive eigenvalues.

For $C^{(g)}=I, C^{(g)^{-\frac{1}{2}}}=I$ and Eq. (28) replicates (22). The transformation $C^{(g)^{-\frac{1}{2}}}$ rescales the step $m^{(g+1)}-m^{(g)}$ within coordinate system given by $B^{(g)}$. Consequently the transformation $C^{(g)^{-\frac{1}{2}}}$ makes the expected length of $p_\sigma^{(g+1)}$ independent of its direction and for any sequence of realized covariance matrices $C_{g=0,1,2,\ldots}^{(g)}$ under random selection $p_\sigma^{(g+1)} \sim N(0.I)$, given $p_\sigma^{(0)} \sim N(0.I)$. To update $\sigma^{(g)}$, $\left\|p_\sigma^{(g+1)}\right\|$ is compared with its expected length $E\|N(0,I)\|$, that is

$$\ln\sigma^{(g+1)}=\ln\sigma^{(g)}+\frac{c_\sigma}{d_\sigma E\|N(0,I)\|}\left(\left\|p_\sigma^{(g+1)}\right\|-E\|N(0,I)\|\right). \tag{23}$$

Here, $d_\sigma \approx 1$ is damping parameter which scales the change magnitude of $\ln\sigma^{(g)}$. The factor $c_\sigma/d_\sigma/E\|N(0,I)\|$ is based on the investigations of the algorithm.

$$E\|N(0,I)\|=\sqrt{2}\Gamma\left(\frac{n+1}{2}\right)/\Gamma\left(\frac{n}{2}\right)\approx\sqrt{n}+O(1/n). \tag{24}$$

It is the expectation of the Euclidean norm of a $N(0,I)$ distributed random vector. The step-size change is unbiased on the log scale because $E[\ln\sigma^{(g+1)}|\sigma(g)] = \ln\sigma^{(g)}$ for $p_\sigma^{(g+1)} \sim N(0,I)$. Because$\sigma^{(g)} > 0$, Eq. (23) is equivalent to

$$\sigma^{(g+1)}=\sigma^{(g)}+\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|p_\sigma^{(g+1)}\right\|}{E\|N(0,I)\|}-1\right)\right). \tag{25}$$

The length of then evolution path is an intuitive and empirically well validated goodness measure for the overall step-length. For $\mu_{eff} > 1$ it is the best measure to the knowledge. Nevertheless, it fails to adapt nearly optimal step-sizes on very noisy objective functions.

3. Classical DE

DE is a simple real-coded evolutionary algorithm. It works through a simple cycle of stages, which are detailed below. In this section we describe the basic operations of DE and introduce necessary notations and terminologies which facilitate the explanation of different adaptive DE algorithms later.

3.1. Initialization of the parameter vectors

DE searches for a global optimum point in a *D*-dimensional continuous hyperspace. It begins with a randomly initiated population of *NP D* dimensional real-valued parameter vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multi-dimensional optimization problem. We shall denote subsequent generations in DE by $G = 0, 1, \ldots, G_{\max}$. Since the parameter vectors are likely to be changed over different generations, we may adopt the following notation for representing the *i*th vector of the population at the current generation:

$$\vec{X}_{i,G} = \left[x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}\right]. \tag{26}$$

For each parameter of the problem, there may be a certain range within which the value of the parameter should lie for better search results. The initial population (at $G = 0$) should cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds:

$$\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\} \quad \text{and } \vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}.$$

Hence we may initialize the *j*th component of the *i*th vector as:

$$x_{j,i,0} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min}), \tag{27}$$

where rand is a uniformly distributed random number lying between 0 and 1 and is instantiated independently for each component of the *i*th vector.

3.2. Mutation with difference vectors

Biologically 'mutation' means a sudden change in the gene characteristics of a chromosome. In the context of the evolutionary computing paradigm, however, mutation is also seen as a change or perturbation with a random element. Most of the real-coded EAs typically simulate the effects of mutation with additive increments, which are randomly generated by a predetermined Probability Density Function (PDF). DE, however, applies a uniform Probability Density Function not to generate increments, but to randomly sample vector differences. In DE, mutation amounts to creating a donor vector $\vec{V}_{i,G}$ for changing each population member $\vec{X}_{i,G}$ (say), in each generation (or in an iteration of the algorithm). In one of the easiest or simplest forms of DE-mutation, to create $\vec{V}_{i,G}$ for each *i*th member of the current population (also called the *target* vector), three other distinct parameter vectors, say the vectors $\vec{X}_{r_1^i}, \vec{X}_{r_{2^i}}, \vec{X}_{r_{3^i}}$ are picked up randomly from the current population. The indices r_1^i, r_2^i, r_3^i are mutually exclusive integers randomly chosen from the range $[1, NP]$, which are also different from the base vector index *i*. These indices are randomly generated once for each mutant vector. Now the difference of any two of these three vectors is scaled by a scalar number *F*and the scaled difference is added to the third one whence we obtain the donor vector $\vec{V}_{i,G}$. We can express the process as,

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot \left(\vec{X}_{r_2^i,n} - \vec{X}_{r_3^i,G}\right). \tag{28}$$

The equation above shows the mutation scheme of the DE which is better designated as DE/rand/1. The following are different mutation strategies used in the literature:

$$(1)\text{“DE/rand/1”} \qquad \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot \left(\vec{X}_{r_2^i,n} - \vec{X}_{r_3^i,G}\right). \tag{29}$$

$$(2)\text{“DE/best/1”} \qquad \vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot \left(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}\right). \tag{30}$$

$$(3)\text{“DE/target-to-best/1”} \qquad \vec{V}_{i,G} = \vec{X}_{i,G} + F \cdot \left(\vec{X}_{best,G} - \vec{X}_{i,G}\right) + F \cdot \left(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}\right). \tag{31}$$

$$(4)\text{“DE/best/2”} \quad \vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot \left(\vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}\right) + F \cdot \left(\vec{X}_{r_3^i,G} - \vec{X}_{r_4^i,G}\right). \tag{32}$$

$$(5)\text{“DE/rand/2”} \qquad \vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot \left(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}\right) + F \cdot \left(\vec{X}_{r_4^i,G} - \vec{X}_{r_5^i,G}\right). \tag{33}$$

The indices $r_1^i, r_2^i, r_3^i, r_4^i$, and r_5^i are distinct integers uniformly chosen from the set $\{1,2,\ldots,NP\}/\{i\}$.These indices are randomly generated once for each donor vector. Again the scaling factor *F* is a positive control parameter for scaling the difference

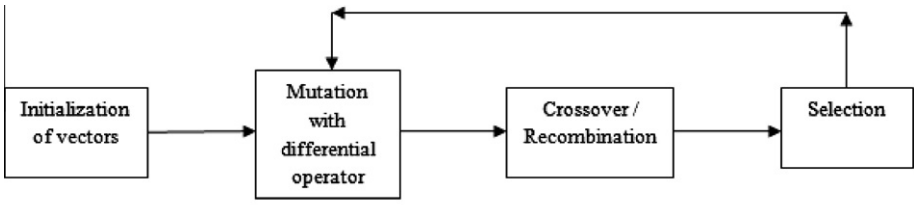

Fig. 1. Illustration of the entire DE algorithm.

vectors. $\vec{X}_{best,G}$ is the best individual vector with the best fitness (i.e. lowest objective function value for minimization problem) in the population at generation G. The general convention used for naming the various mutation strategies is DE/x/y/z, where DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed and y is the number of difference vectors considered for perturbation of x. z stands for the type of crossover being used (exp: exponential; bin: binomial). The following section discusses the crossover step in DE.

3.3. Crossover

A crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges its components with the target vector $\vec{X}_{i,G}$ under this operation to form the final trial/offspring vector $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, \ldots, u_{D,i,G}]$. The scheme may be outlined as:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } rand_{i,j}[0,1] \leqslant Cr \quad \text{or } j = j_{rand} \\ x_{j,i,G}, & \text{otherwise,} \end{cases} \tag{34}$$

where, as before, $rand_{i,j}[0,1]$ is a uniformly distributed random number, which is called anew for each jth component of the ith parameter vector and Cr is called the crossover rate that appears as an important control parameter of DE. Cr is only approximating the true probability p_{Cr} of the event that a component of the trial vector will be inherited from the donor. $j_{rand} \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component from $\vec{V}_{i,G}$.

3.4. Selection

The next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation, i.e. at $G = G + 1$. The selection operation is described as:

$$\vec{X}_{i,G+1} = \vec{U}_{i,G}, \quad \text{if} \quad f\left(\vec{U}_{i,G}\right) \leqslant f(\vec{X}_{i,G}) = \vec{X}_{i,G}, \quad \text{if } f(\vec{U}_{i,G}) > f\left(\vec{X}_{i,G}\right), \tag{35}$$

where $f(\vec{X})$ is the objective function to be minimized. Note that throughout the article, we shall use the terms *objective function value* and *fitness*interchangeably. But, always for minimization problems, a lower objective function value will correspond to higher fitness. The entire cycle of stages comprising the DE algorithm is presented in Fig. 1 below.

4. The proposed algorithm DCMA-EA

In our proposed hybridization scheme DCMA-EA the differential mutation, binomial crossover and selection operations of DE have been incorporated into the structure of a CMA-ES algorithm in order to achieve a better performance than both of the ancestors. The CMA-ES is a stochastic method for real-parameter (continuous domain) optimization of non-linear, non-convex functions. It takes the help of adaptation of covariance matrix which is subjected into a multivariate normal distribution whose search distribution is hugely utilized to detect the landscape of a convex-quadratic objective function. The objective of covariance matrix adaptation is to approximate the inverse Hessian matrix, similar to a quasi-Newton method to the covariance matrix. More general, the objective is to suit the search distribution to the contour lines of the objective function. Further, to increase the robustness of the population, the binomial crossover mechanism is introduced as it induces exchange of body-parts, i.e. components within the population agents. Finally, to increase the exploitative nature of DCMA-EA the greedy selection procedure of DE is implemented that promotes better individuals to the next generation.

4.1. Introduction of mutation

The traditional CMA-ES algorithm which basically bears the same analogy of the gradient based Quasi-newton method uses the adaptation of the covariance matrix to identify the function landscape which is a convex-quadratic one through the concept of Hessian matrix. What it does is that it sets the covariance matrix of the search distribution to the inverse

Hessian matrix which is equivalent to rescaling the ellipsoid projection of the multivariate normal distribution into a spherical one. For a particular function landscape if it becomes qualified to convert it into a spherical projection by correspondingly adapting the eigen-decomposed matrix (B,D) then the algorithm converges fully on the global optima of a given problem. But the main hiccups arrives when more complex functions comes into account like noisy and hybrid composition functions then it fails to detect its landscape so as to adapt the eigen-decomposed matrices for its multivariate normal distribution to detect its function structure and the algorithm becomes liable to be trapped in a local optima. That is the main reason of the traditional CMA-ES failing horribly in the noisy and hybrid functions. In order to remove this major demerit of the CMA-ES a new differential perturbation scheme has been introduced in this proposed algorithm. In the CMA-ES algorithm the new population vector is created by this following equation.

$$\vec{X}_{i,g} = m + \sigma \cdot N(0,C) = m + \sigma \cdot B \cdot D \cdot randn(\dim)^T, \tag{36}$$

where *randn*(dim) is a collection of random numbers taken from a normal distribution of zero mean and standard deviation 1, and having elements equal to the dimension of the function in hand. The determination of m and the evolution of σ has been discussed earlier in Section 2.

To overcome the limitations of the CMA-ES algorithm discussed above, a difference vector based DE mutation factor is incorporated followed by the addition of the controlled step-size generated by the co-variance matrix adaptation. In the context of EAs, mutation is treated as a random change of some parameter. Real parameter EAs typically simulate the effects of mutation with additive increments that are randomly generated by a predefined and fixed Probability Density Function (PDF). DE differs markedly from algorithms like ES and EP in consideration of the fact that it mutates the base vectors (secondary parents) with scaled population-derived difference vectors. As generations pass, these differences tend to adapt to the natural scaling of the problem. For example, if the population becomes compact in one variable but remains widely dispersed in another, the difference vectors sampled from it will be small in the former variable, yet large in the latter. This automatic adaptation significantly improves the convergence performance of the algorithm. Moreover, according to the analytical studies undertaken by Zharie [53], the explorative power of DE is also greater as compared to ES and accounts for its better performance significantly. These facts motivated us to use DE-type mutation in order to mitigate the problems in CMA-ES. In DCMA-EA the mutated vectors are generated by using a controlled share of the target particle and the population mean followed by the addition difference vector-based mutation factor and the controlled step-size as shown in the following equation below:

$$\vec{V}_{i,g} = m \cdot (1-P) + P \cdot \vec{X}_{i,g} + F \cdot (\vec{X}_{r_1^i,g} - \vec{X}_{r_2^i,g}) + (1-P) \cdot \sigma \cdot B \cdot D \cdot randn(\dim)^T, \tag{37}$$

where $\vec{X}_{r_1^i,g}$ and $\vec{X}_{r_2^i,g}$ are vectors chosen randomly from the current population. m is mean of the current population vectors. P is a typical control parameter which controls the participation of the mean vector of the current population and target vectors. F is the scale factor of Differential Evolution. σ is standard deviation of current population vectors. The symbols B and D have been discussed earlier.

4.2. Randomized scale factor

The scale factor F is an important control parameter that scales the difference vectors in Differential Evolution. If the value of F is kept a constant value the diversification of the system is lost as all the particles are perturbed by the same difference vector component. To overcome this drawback the scale factor F_i for each target vector is generated according to a uniformly distributed random number between 0.5 and 1 in each generation as shown below:

$$F_i = 0.5 + 0.5 \cdot rand(0,1) \tag{38}$$

where $rand(0,1)$ is a uniformly distributed random number within the range $[0,1]$. This permits random variations in the scaling of the difference vector allowing the individuals of the system to sample diverse zones of the search space enhancing the explorative power during the early stages of the search. At the later stages when majority of the population agents point towards the interior of the region in which the suspected global optima lies, due to the stochastically scaled difference vector the donor generated has a moderate chance of jumping to a better location in the multimodal search space lowering the probability to get trapped in a local optima. The utility for including a certain degree of randomization while choosing the values of F for DE is explicitly reported in [38].

4.3. Significance of the control parameter P

The control parameter P in DCMA-EA has been dynamically increased from 0.5 to 1 with generation as shown in the equation below:

$$P = 0.5 \cdot \left(1 + \frac{iter}{iter\_\max}\right). \tag{39}$$

The main reason behind this control parameter is to set a trade-off between the mean and the target vector in the mutation strategy. At every stage of the algorithm performance it requires a proper trade-off between explorative and exploitative

power. Using only the mean vector in mutation does not help the cause because all the vectors then gets perturbed about only the same mean vectors which not only reduces the explorative power of the algorithm in first stage but also hampers the performance in the later stage due to the resultant reduced population diversity where the vectors need to avoid the traps in local optima of multimodal landscape. But when a proper percentage of the mean vector and the current target vectors are used there are some benefits. Incorporation of the target vector helps the generated offspring to keep track of parent vectors to suitably set the evolution path and also it wipes out the stagnancy of the population because each vector gets mutated by its original parent vector, not only the mean vector. So, this variation of the control parameter helps the vectors at later stages of the algorithm performance by eliminating the possibilities of the vectors to get stagnated in local optima as it compels the target vector to take part more in mutation at the higher generations.

4.4. Introduction of crossover

To avoid premature convergence at local optima, there is a need to enhance the potential diversity of the population. To circumvent this defect we have implemented the normal DE crossover mechanism after generating the donor vectors through mutation. The basic objective is component injection, i.e. the information contained in the parents is incorporated into the mutated vectors to generate the corresponding trial vectors or offspring. The DE family of algorithms can use two kinds of crossover methods – exponential (or two-point modulo) and binomial (or uniform). In this article we focus on the widely used binomial crossover that is performed on each of the D variables whenever a randomly generated number between 0 and 1 is less than or equal to the Cr value. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution. The entire scheme is discussed above in section 3.3.

4.5. Crossover probability determination

At every generation, the crossover probability Cr_i for each mutated vector is independently generated in accordance with a normal distribution of mean Cr_m and standard deviation 0.1 as shown below:

$$Cr = Gaussian(Cr_m, 0.1). \tag{40}$$

Now a value of Cr generated from the distribution described above may be greater than 1. Those Cr values are again regenerated. The value of C_{rm} is set to be a slightly higher value (0.9) as it favours non-separable functions which exist largely in real world optimization field. The normal distribution is perfect for the crossover probability values as it has a short tail property. This property does not permit vary large values of Cr greater than 1 which ultimately would have to be truncated to unity. A unity value of Cr means that no component of the parent vector is inherited into the offspring which hinders the basic mechanism of DE.

4.6. Selection

In EAs exploitation is done via selection that promotes better individuals to the subsequent generation. To increase the exploitative power of DCMA-EA, the selection procedure of DE is incorporated in the structure of the hybrid algorithm. The basic selection procedure is the traditional competitive selection in which individuals are ranked according to their fitness. Next, fittest particles are transferred to the next iteration and those with lower fitness are discarded. But the major shortcoming of this kind of selection is that it ignores the inferior solutions. But in many optimization problems it is seen that the less fit particles may provide useful information about the promising progress direction.

The alternative choice to competitive selection is the one-to-one greedy selection procedure of DE which overcomes this defect by giving preference to the particles with inferior fitness. In this process the fitness of the parents are compared with those of the offspring and the fitter individuals are passed onto the consequent generations. The entire selection operation is outlined in Section 3.4. Therefore, if the new trial vector/offspring yields an equal or lower value of the objective function, it replaces the corresponding target (parent) vector in the next generation; otherwise the target is retained in the population. Hence, the population either gets better (with respect to the minimization of the objective function) or remains the same in fitness status, but never deteriorates improving the convergence performance as well as the exploitative power of the proposed hybrid algorithm.

4.7. Parameter settings in DCMA-EA

In the original CMA-ES algorithm the population size is dependent on problem dimensionality. But in DCMA-EA the population size is kept constant for the DE mutation and crossover to perform successfully on a large number of vectors. The population size has been kept fixed at 50 for all the problem dimensions considered here. The remaining parameters are initialized as follows.

1. $Cr_m = 0.9$
2. $\mu = \frac{pop\_size}{2}$

3. $w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j},\ w'_i = \ln(\mu + 0.5) - \ln i\ \forall i(1)\mu$
4. $\mu_{eff} = \frac{\left(\sum_{i=1}^{\mu} w_i\right)^2}{\sum_{i=1}^{\mu} w_i^2}$
5. $c_\sigma = \frac{\mu_{eff}+2}{\dim+\mu_{eff}+5}$
6. $d_\sigma = 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_{eff}-1}{\dim+1}} - 1\right) + c_\sigma$
7. $c_c = \frac{4+\frac{\mu_{eff}}{\dim}}{\dim+4+\frac{2\cdot\mu_{eff}}{\dim}}$
8. $c_1 = \frac{2}{(\dim+1.3)^2+\mu_{eff}}$
9. $c_\mu = \min\left(1 - c_1, 2 \cdot \frac{\mu_{eff}-2+\frac{1}{\mu_{eff}}}{(\dim+2)^2+\mu_{eff}}\right).$

where the symbol *pop_size* represents the population size and dim symbolizes the problem dimension.

4.8. Putting it all together

A complete pseudo-code of the hybrid DCMA-EA algorithm is presented in Table 1 below.

4.9. Algorithmic rationale and functioning behind DCMA-EA

CMA-ES has emerged as a very competitive real-parameter optimizer for continuous search spaces. The strong point of CMA-ES compared to other evolutionary algorithms lies in its invariance against the linear transformation. Consequently CMA-ES has performed excessively good in functions exhibiting linear transformations. For a given objective function search space if CMA-ES is able to rescale the ellipsoid projection of the multivariate normal distribution into a spherical one then it converges successfully to the desired optima for that problem. But the main hiccups arise when we try to optimize more complex landscapes such as noisy and hybrid composition functions with CMA-ES. The reason behind this is that CMA-ES fails to rescale more complex landscapes into a spherical one and thus is unable to converge in those landscapes as evident from the poor performance of CMA-ES in noisy and hybrid composition functions.

In our proposed hybridization scheme, our aim is to improve the performance of CMA-ES over a wide range of objective functions. Now, a difference vector based DE mutation operator is available that has the capability to adapt to the natural scaling of any fitness landscape. Consequently DE performs substantially well over a wide range of problems. Our idea is to incorporate the difference vector based mutation operator of DE into the CMA-ES mutation scheme so that CMA-ES has the power to acclimatize with more complex landscapes leading to successful convergence performance. That is the main algorithmic rationale behind this hybridization. The hybrid algorithm DCMA-EA as described in the previous sections improves the performance of CMA-ES to a greater extent in case of noisy, hybrid composition functions and even some real world optimization problems as evident from the Sections 5 and 6.

4.10. DCMA-EA: relation to memetic computing

Since late 1990s, an algorithmic philosophy, going under the name of memetic computing (MC) [4], started to gain a wide popularity among the researchers working on optimization problems. MC is centered on the idea of problem solving with the notion of "memes" (a term coined by Dawkins [8]) embedded in a computational framework. Memes can be loosely defined as the basic units of culture that can be transmitted across generations by non-genetic means, e.g. via imitation. MC began with the definition of the concept of MAs, which basically hybridize a deterministic local search with some global evolutionary optimizer like genetic algorithms (GAs), DE, etc. for solving a given optimization problem. The basic objective of MAs is to enable the individuals to pass over the knowledge (i.e. memes) to other individuals through a process of life-time learning [41], besides transmitting the genetic information to the offspring through the evolutionary operators (like mutation, recombination, and selection).

In recent past, a few significant MAs using DE as the core global optimizer have surfaced. Neri and Tirronen [39] proposed a DE-based MA that employs two local search algorithms within a self-adaptive scheme. The authors called this algorithm the Scale Factor Local Search DE (SFLSDE). The used local search schemes aim at detecting a value of the scale factor F corresponding to an offspring with a higher fitness, while the generation is executed. In [49], Tirronen et al. came up with a DE-based MA employing three local search algorithms coordinated by means of fitness diversity adaptation and a probabilistic scheme for designing digital filters that can detect defects of the paper produced during an industrial process. In [3], Caponio et al. incorporated PSO and two other LS algorithms (Nelder mead algorithm and Rosenbrock algorithm) in the framework of DE. Neri and Mininno [37] proposed a memetic compact DE to solve a robotics based optimization problem that needs to be solved without the aid of a full-power computing device. Similarly, recently Molina et al. [34] came up with an MA that is based on local search chains and uses CMA-ES as one of its intense local search components. Neri et al. [40] recently proposed a memetic computing approach based on the compact DE algorithm for solving real-parameter optimization problems characterized by memory limitations.

Table 1
Pseudo-code of DCMA-EA.

Begin
Initialize the parameters pop_size, μ, $w_i \forall\ 1,\ldots,\mu$, μ_{eff}, c_σ, d_σ, c_c, c_1, c_μ as per the Section 4.7
Set the initial evolution paths $p_\sigma = p_c = 0$, $C = I, B = I_n$, $D = ones(D,1)$ elements of m randomly from 0–1, $\sigma = 0.5$
Create the initial population by the following equation
For $i = 1$ to *pop_size*

$$\vec{X}_i = m + \sigma \cdot B \cdot D \cdot randn(D)^T$$

end
For g=1 to G
m is updated by Eq. (8) after sorting the population
p_σ and p_c is updated by Eqs. (22) and (19)
C is updated by Eq. (21)
σ is updated by the Eq. (25)
Obtain B, D through eigen decomposition of C
Obtain $C^{-\frac{1}{2}}$ from Eq. (5)
Obtain P by the Eq. (39)
Obtain F by the Eq. (38)
For $i = 1$ to *pop_size*
Randomly choose $\vec{X}_{r_1^i,g}, \vec{X}_{r_2^i,g}$ from current population by method of rotating index

$$\vec{V}_{i,g} = m \cdot (1-P) + P \cdot \vec{X}_{i,g} + F \cdot \left(\vec{X}_{r_1^i,g} - \vec{X}_{r_2^i,g}\right) + (1-P) \cdot \sigma \cdot B \cdot D \cdot randn(D)^T$$

End For
For $i = 1$ to NP
Generate j_{rand}=ceil (rand (1,D))
For$j = 1$ to D
Generate Cr according to the Eq. (40)
If $rand_{i,j}[0,1] \leqslant Cr$ or $j = j_{rand}$

$$u_{j,i,G} = v_{j,i,G}$$

Else

$$u_{j,i,G} = x_{j,i,G}$$

End If
End For
If $f(\vec{U}_{i,G}) \leqslant f(\vec{X}_{i,G})$

$$x_{i,G+1} = u_{i,G}$$

Else

$$x_{i,G+1} = x_{i,G};$$

End If
End for
End for
End

Based on the discussions in last two paragraphs, the proposed DCMA-EA may not be categorized as an MA in strict sense as it does not embed any deterministic local refinement procedure. Nevertheless, DCMA-EA can be well posed as a more generalized memetic computing approach as it has the objective of devising a hybrid of two evolutionary global optimizers such that the hybrid algorithm may significantly outperform these two techniques separately, and this is certainly an objective of the memetic computing framework [20].

5. Experimental setup and results

5.1. Benchmark functions used

In order to experimentally illustrate the effectiveness of the proposed hybrid algorithm, we choose a standard test bench comprising of 20 numerical benchmarks of different characteristics such as multimodality, multiplicative noise in fitness, shift in dimensionality, function rotation as well as composition of functions. A brief description of the functions is given in Table 2 below. Functions f_1–f_5 are taken from the test-suite provided in [52] while rest of the functions belong to the CEC 2005 benchmark function set [27,47]. Functions f_1–f_4 and f_6 are uni-modal and rest of the problems are multimodal in nature.

Table 2
Mathematical test functions used in this study, including a short description of their individual characteristics.

Serial no.	Function name	Function description	Search range
f_1	Sphere function	$f(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$
f_2	Schwefel 2.22 function	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]$
f_3	Schwefel 1.2 function	$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]$
f_4	Schwefel 2.21 function	$f(x) = \max_i \{ \lvert x_i \rvert \}$	$[-100, 100]$
f_5	Rosenbrock's function	$f(x) = \sum_{i=1}^{D-1} \left[100 \cdot \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	$[-30, 30]$
f_6	Shifted Schwefel's Problem 1.2 with noise in fitness	Corresponds to f_4 of [47]	$[-100, 100]$
f_7	Shifted rotated Griewank's function without bounds	Corresponds to f_7 of [47]	$[-600, 600]$
f_8	Shifted rotated Ackley's function with global optimum on bounds	Corresponds to f_8 of [47]	$[-32, 32]$
f_9	Shifted Rastrigin's function	Corresponds to f_9 of [47]	$[-5, 5]$
f_{10}	Shifted rotated Rastrigin's function	Corresponds to f_{10} of [47]	$[-5, 5]$
f_{11}	Expanded rotated extended Scaffe's F6	Corresponds to f_{14} of [47]	$[-100, 100]$
f_{12}	Hybrid composition function 1	Corresponds to f_{15} of [47]	$[-5, 5]$
f_{13}	Rotated hybrid composition function 1	Corresponds to f_{16} of [47]	$[-5, 5]$
f_{14}	Rotated hybrid composition function 1 with Noise in Fitness	Corresponds to f_{17} of [47]	$[-5, 5]$
f_{15}	Rotated hybrid composition function 2	Corresponds to f_{18} of [47]	$[-5, 5]$
f_{16}	Rotated hybrid composition function 2 with a narrow basin for the global optimum	Corresponds to f_{19} of [47]	$[-5, 5]$
f_{17}	Rotated hybrid composition function 2 with the global optimum on the bounds	Corresponds to f_{20} of [47]	$[-5, 5]$
f_{18}	Rotated hybrid composition function 3	Corresponds to f_{21} of [47]	$[-5, 5]$
f_{19}	Rotated hybrid composition function 4	Corresponds to f_{24} of [47]	$[-5, 5]$
f_{20}	Rotated hybrid composition function 4 without bounds	Corresponds to f_{25} of [47]	$[-2, 5]$

5.2. Algorithms compared

In order to validate the potentiality of the incorporation of DE-based operators in CMA-ES, we compare its performance with the ancestor algorithms CMA-ES and DE/rand/1/bin. To judge how much improvement we achieve over the recently invented renowned DE and CMA-ES variants, we compare DCMA-EA with the Self-Adaptive DE (SaDE), an adaptive DE with DE/current- to-*p*best/1 mutation scheme and optional external archive (JADE) and the restart CMA-ES with increasing population size (IPOP-CMA-ES). Finally we also compare DCMA-EA with a very powerful and real parameter optimizer of current interest: the DMS-PSO (Dynamic Multi Swarm Particle Swarm Optimizer).

We follow the parameter settings in the original paper of SaDE and DMS-PSO. The parameters of DE/rand/1/bin are set to be $F = 0.9$ and $Cr = 0.9$ as recommended in the original literature. For JADE, we set the parameters to be fixed: $p = 0.05$ and $c = 0.1$. For all the contestant algorithms we choose the best suited parametric set-up chosen with guidelines from their respective literatures.

5.3. Results on numerical benchmarks

Tables 3 and 4 show the mean and the standard deviation of the best-of-run errors for 50 independent runs of each of the six algorithms on 20 numerical benchmarks for dimensions 30 and 50, respectively. Note that the best-of-the-run error corresponds to absolute difference between the best-of-the-run value $f(\vec{x}_{best})$ and the actual optimum f^* of a particular objective function, i.e. $|f(\vec{x}_{best}) - f^*|$. For 30-D and 50-D problems the maximum number of Function Evaluations is set to be 3e+05 and 5e+05, respectively. For function f_9, we set $Cr_m = 0.1$ as it is a separable function. The control parameter P for the functions f_1–f_5 has been kept at a low value (0.2) as for these functions CMA-ES performs better than DE. To enhance the performance further, only the effect of difference vector-based mutation is incorporated.

In order to establish the statistical significance of the results, a non-parametric statistical test called Wilcoxon's rank sum test for independent samples is conducted where the confidence level has been fixed to 0.95. Tables 5 and 6 summarize the results of the rank sum test for each version of DCMA-EA against its corresponding population-based algorithm. The '+' symbol denotes that the mean best-of-run result obtained with DCMA-EA is statistically significantly better than its corresponding competitor algorithm over the concerned benchmark function. The '=' indicates that the mean errors yielded by DCMA-EA does not differ significantly from that produced by the competitor algorithm. Finally a '−' symbol means that DCMA-EA produced statistically inferior mean error in comparison to its fellow algorithm for the corresponding cell of the table.

Table 3 indicates that considering the mean of error values for 30D problems, DCMA-EA outperforms all the contestant algorithms in a statistically significant fashion (as revealed by Table 5) over 17 functions. It managed to remain second best for function f_8 being outperformed by IPOP-CMA-ES alone. On function f_{18} the performance of DCMA-EA remained comparable to those of IPOP-CMA-ES, the former, however, achieving lowest standard deviation once again. A close scrutiny of Table 4 reveals that the performance of DCMA-EA was not affected in a worse way with the enhancement of search-space

Table 3
Experimental results of 30-dimensional problems f_1–f_{20}, averaged over 50 independent runs.

Functions	DE/rand/1/bin Mean (std. dev.)	SaDE Mean (std. dev.)	JADE Mean (std. dev.)	DMS-PSO Mean (std. dev.)	CMA-ES Mean (std. dev.)	IPOP-CMA-ES Mean (std. dev.)	DCMA-EA Mean (std. dev.)
f_1	9.8154e−30	4.5263e−36	1.3287e−72	1.3872e−62	4.346e−233	6.48e−260	**3.0221e−293**
	(5.6276e−29)	(6.7834e−36)	(9.2543e−72)	(2.7583e−62)	(8.7456e−233)	(8.98e−260)	(**9.8752e−294**)
f_2	1.6243e−18	1.9675e−23	3.9845e−32	9.3734e−30	2.993e−102	4.58e−119	**7.4940e−135**
	(2.3453e−18)	(1.0547e−23)	(2.7345e−31)	(6.7324e−30)	(1.324e−102)	(9.48e−119)	(**6.3575e−135**)
f_3	6.6243e−10	9.0145e−20	6.0198e−65	2.5643e−23	4.813e−125	3.49e−140	**1.5406e−165**
	(8.4326e−10)	(5.4321e−19)	(1.9432e−84)	(3.9564e−22)	(2.364e−125)	(8.58e−140)	(**8.0152e−166**)
f_4	1.0134e+00	7.4123e−07	4.3123e−45	4.4234e−18	1.143e−72	9.84e−88	**1.0751e−98**
	(1.123e+00)	(1.823e−06)	(1.2344e−44)	(9.342e−18)	(2.435e−72)	(3.67e−88)	(**6.9439e−99**)
f_5	2.134e+00	2.1243e+01	3.2976e−01	4.784e−01	4.702e−01	1.93e−01	**4.9834e−02**
	(1.523e+00)	(7.8254e+00)	(1.1254e+00)	(1.322e+00)	(1.234e+00)	(1.06e+00)	(**1.0143e+00**)
f_6	5.04e−01	5.8160e−04	5.1159e−05	2.5487e+03	1.3840e+04	1.11e+04	**6.3702e−09**
	(8.58e−01)	(1.4479e−04)	(4.0194e−05)	(3.0638e+02)	(6.4478e+03)	(3.02e+04)	(**7.1254e−09**)
f_7	9.65e−01	8.2727e−03	6.9598e−03	6.9990e−03	6.8843e−03	5.31e−12	**1.1102e−20**
	(9.14e−02)	(1.1445e−02)	(4.4845e−03)	(4.5371e−03)	(2.8731e−02)	(1.41e−12)	(**1.1845e−20**)
f_8	2.095e+01	2.094e+01	2.095e+01	2.093e+01	2.094e+01	**2.01e+01**	2.0903e+01
	(6.008e−02)	(6.041e−02)	(5.351e−02)	(5.288e−02)	(4.480e−02)	(**2.79e−01**)	(5.4523e−02)
f_9	4.3742e+01	4.179e−01	**3.0313e−08**	1.759e+01	2.8723e+01	9.38e−01	1.4890e+01
	(9.9346e+00)	(1.156e+00)	(**8.3551e−09**)	(3.022e+00)	(7.8124e+00)	(1.18e+00)	(9.4523e+00)
f_{10}	6.16e+01	3.5758e+01	3.0313e+01	3.7410e+01	4.8723e+01	**1.65e+00**	4.9024e+01
	(4.56e+01)	(6.0809e+00)	(8.3551e+00)	(5.2883e+00)	(9.8124e+00)	(**1.35e+00**)	(1.1523e+01)
f_{11}	1.3372e+01	1.2860e+01	1.2771e+01	1.2710e+01	1.3264e+01	1.29e+01	**1.2408e+01**
	(3.8475e−01)	(3.0809e−01)	(4.3551e−01)	(5.2883e−01)	(4.8124e−01)	(4.39e−01)	(**3.4523e−01**)
f_{12}	4.8443e+02	3.2775e+02	3.3284e+02	3.4464e+02	7.2856e+02	3.3546e+02	**3.12e+02**
	(2.1412e+01)	(9.6450e+01)	(9.0503e+01)	(5.0662e+01)	(3.9823e+01)	(8.5436e+01)	(**2.6786e+01**)
f_{13}	2.8293e+02	1.3795e+02	1.0238e+02	1.1950e+02	2.2543e+02	1.06e+02	**9.2307e+01**
	(1.1875e+001)	(1.7023e+01)	(3.9432e+01)	(1.206 8e+01)	(1.9832e+01)	(2.37e+01)	(**3.8872e+01**)
f_{14}	3.0965e+02	1.5092e+03	1.0619e+02	1.1950e+02	2.2543e+02	2.91e+02	**9.9328e+01**
	(1.5871e+01)	(9.3631e+02)	(3.5763e+01)	(1.206 8e+02)	(1.9832e+01)	(1.93e+02)	(**3.9757e+01**)
f_{15}	9.1344e+002	9.5447e+02	8.9579e+02	9.1053e+02	9.0441e+02	9.04e+02	**8.6726e+02**
	(8.43e−001)	(3.4387e+01)	(1.6523e−01)	(1.5761e+00)	(2.8822e−01)	(2.88e−01)	(**1.5209e−01**)
	9.1331e+002	8.4583e+02	8.7644e+02	9.1060e+02	9.0431e+02	9.04e+2	**8.4325e+02**
f_{17}	(1.21e+000)	(6.2155e+01)	(1.5419e−01)	(1.3383e+00)	(2.7115e−01)	(2.71e−01)	(**1.5340e−01**)
	9.1332e+002	2.0402e+03	9.1697e+02	9.0189e+02	9.0406e+02	9.04e+02	**8.1626e+02**
	(1.16e+000)	(8.7683e+02)	(1.7231e−01)	(3.0719e+01)	(2.4837e−01)	(2.48e−01)	(**2.6786e−01**)
f_{18}	5.8135e+02	1.730e+03	8.5838e+02	8.635e+02	8.764e+02	**5.000e+02**	**5.0000e+02**
	(2.6247e+01)	(5.118e+02)	(1.1013e+01)	(3.015e+01)	(1.311e+01)	(**1.31e−13**)	(**0**)
f_{19}	3.1434e+02	2.0985e+02	2.1141e+02	2.1074e+02	9.4326e+02	9.10e+02	**2.0893e+02**
	(3.2249e+01)	(2.0882e+00)	(1.5664e+00)	(1.015e+00)	(1.3119e+02)	(1.48e+02)	(**1.6302e+00**)
f_{20}	7.86e+02	5.001e+02	2.1123e+02	9.889e+02	5.000e+02	2.1079e+02	**2.0986e+02**
	(2.17e+01)	(5.683e−02)	(4.2936e+00)	(3.015e+01)	(1.311e+04)	(1.311e+00)	(**1.6271e+00**)

dimensionality. It achieved statistically superior performance compared to all other competitor algorithms for 17 benchmark instances in 50*D* problems as revealed by Table 6. It has managed to remain second best for function f_8 being outperformed by IPOP-CMA-ES alone. Note that DCMA-EA has shown its superiority in case of rotated, hybrid as well as functions with noise in fitness. So, function rotation, incorporating multiplicative noise in them as well as composition of functions does not hamper the performance of the proposed hybrid algorithm significantly.

From the above results it is clearly visible that by embedding the DE-based operators in CMA-ES, huge improvement in performance has been achieved in case of noisy and hybrid composition functions. The strength of DCMA-EA lies in the fact that it has shown good performance over the entire range of benchmark functions considered here except a few. The superior performance of DCMA-EA can be attributed to the proper sharing of the DE and CMA-ES mutation schemes which, in turn, depends on the proper choice of the control parameter *P*. The dependence of DCMA-EA on control parameter *P* can be ascribed as a weak point of this hybrid scheme.

For further illustration, we plot the convergence graph for the median run of (where the runs were sorted according to the final best error values achieved in each) four benchmarks in 30D as shown in Fig. 2. As evident from the convergence characteristics in Fig. 2, the overall convergence speed of DCMA-EA is the best among the contestant algorithms. We restrained from giving all the graphs in order to save space.

5.4. Parametric study of DCMA-EA

In this section we are going to explain the rationale behind the choice of parameters in DCMA-EA. DCMA-EA consists of a particular set of parameters among which μ, w_i, μ_{eff}, c_σ, d_σ, c_c, c_1 and c_μ are the inherent parameters of CMA-ES and therefore they are not the control parameters of DCMA-EA as they are not associated with the hybridization process.

Table 4
Experimental results of 50-dimensional problems f_1–f_{20}, averaged over 50 independent runs.

Functions	DE/rand/1/bin Mean (std. dev.)	SaDE Mean (std. dev.)	JADE Mean (std. dev.)	DMS-PSO Mean (std. dev.)	CMA-ES Mean (std. dev.)	IPOP-CMA-ES Mean (std. dev.)	DCMA-EA Mean (std. dev.)
f_1	3.5541e−07 (9.6861e+00)	2.9534e−25 (3.2918e−08)	5.4624e−67 (1.8266e−66)	6.9643e−23 (4.8642e−23)	4.2836e−195 (3.2646e−195)	7.98e−212 (4.85e−212)	**9.0381e−222** (**7.5662e−222**)
f_2	2.3142e−06 (5.6982e−01)	1.7816e−32 (3.8028e−06)	9.2018e−60 (2.2836e−50)	2.8534e−25 (1.3625e−03)	8.2838e−92 (2.4737e−92)	4.87e−138 (8.25e−138)	**5.6446e−160** (**6.8592e−160**)
f_3	7.7352e−01 (8.4326e−10)	3.9275e−17 (5.4321e−19)	5.6545e−32 (1.9432e−84)	6.5142e−15 (3.9564e−22)	6.0184e−112 (7.3747e−112)	6.74e−146 (3.86e−146)	**9.7254e−161** (**5.0382e−161**)
f_4	3.8654e+01 (1.123e+00)	3.7254–10 (1.823e−06)	9.1743–21 (1.2344e−44)	6.4134e−01 (9.342e−18)	5.1143e−25 (2.435e−25)	1.74e−42 (9.25e−43)	**4.0051e−55** (**6.9439e−99**)
f_5	1.134e+01 (1.523e+00)	3.1243e+01 (7.8254e+00)	9.2976e−01 (1.1254e+00)	8.784e−01 (1.322e+00)	9.8264e−01 (1.3932e+00)	3.57e−01 (1.02e+00)	**1.2274e−01** (**1.0325e**+00)
f_6	1.180e+04 (3.332e+03)	9.778e+01 (9.835e+01)	3.160e−01 (4.134e−01)	1.0938e+02 (2.2454e+01)	4.4217e+04 (2.6463e+04)	4.68e+05 (3.11e+05)	**7.8127e−02** (**6.3737e−03**)
f_7	6.195e+03 (8.237e−01)	6.195e+03 (4.594e−01)	6.193e+03 (1.840e+00)	6.195e+03 (4.9873e−01)	7.5421e−03 (9.2636e−03)	3.22e−03 (1.03e−03)	**2.8422e−14** (**1.3721e−14**)
f_8	2.114e+01 (3.330e−02)	2.113e+01 (3.458e−02)	2.114e+01 (3.251e−02)	2.113e+01 (3.918e−02)	2.113e+01 (4.841e−02)	**2.04e+01** (**3.25e−01**)	2.1097e+01 (2.3834e−02)
f_9	3.468e+02 (1.199e+01)	6.148e+00 (1.266e+01)	3.352e+01 (2.591e+00)	7.620e+01 (1.706e+01)	1.406e+02 (2.939e+01)	**1.39e+00** (**1.11e+00**)	1.7523e+02 (3.475e+01)
f_{10}	3.763e+02 (1.578e+01)	6.342e+01 (1.287e+01)	1.935e+02 (2.060e+01)	1.856e+02 (2.9184e+01)	1.0148e+02 (1.2836e+01)	**1.72e+00** (**1.42e+00**)	1.9894e+02 (3.0834e+01)
f_{11}	2.339e+01 (1.486e−01)	2.284e+01 (1.803e−01)	2.284e+02 (2.983e−01)	2.262e+01 (1.375e−01)	2.309e+01 (2.843e−01)	2.29e+01 (5.78e−01)	**2.236e+01** (**4.172e−01**)
f_{12}	5.8342e+02 (2.1412e+01)	4.2775e+02 (8.6524e+01)	4.1972e+02 (9.0503e+01)	4.4710e+02 (5.0662e+01)	7.2856e+02 (3.9823e+01)	4.36e+02 (9.36e+01)	**4.0311e+02** (**6.8374e+01**)
f_{13}	4.9374e+002 (1.1875e+001)	1.5798e+02 (5.9172e+01)	1.2173e+02 (3.9432e+01)	2.8550e+02 (1.206 8e+02)	3.7505e+02 (4.9832e+01)	1.32e+02 (2.93e+01)	**1.1630e+02** (**2.7272e+01**)
f_{14}	4.0143e+002 (1.5871e+01)	1.5053e+03 (8.1635e+02)	3.8162e+02 (3.6534e+01)	2.8172e+02 (1.206 8e+02)	3.2543e+02 (1.9832e+01)	2.34e+02 (1.58e+02)	**1.9328e+02** (**1.9163e+02**)
f_{15}	9.9018e+002 (8.43e−001)	9.7412e+02 (5.8163e+01)	1.0579e+03 (4.9173e−01)	9.6253e+02 (1.5761e+00)	9.8441e+02 (2.8822e−01)	9.78e+02 (8.42e−01)	**9.1326e+02** (**1.6453e−01**)
f_{16}	9.7456e+002 (1.21e+000)	8.6565e+02 (1.2473e+01)	9.8755e+02 (1.5419e−01)	9.9173e+02 (1.3383e+00)	9.3431e+02 (2.7115e−01)	9.12e+02 (7.26e−01)	**8.5325e+02** (**1.1364e−01**)
f_{17}	1.0332e+03 (1.16e+000)	2.3487e+03 (1.2745e+02)	9.8697e+02 (1.7231e−01)	9.8889e+02 (3.0719e+01)	9.8406e+02 (2.4837e−01)	9.76e+02 (7.26e−01)	**9.1206e+02** (**2.9163e−01**)
f_{18}	1.004e+03 (1.101e+00)	9.903e+02 (1.645e+02)	8.523e+02 (2.330e+02)	9.770e+02 (1.306e+02)	1.039e+03 (2.096e+01)	1.00e+03 (8.19e−01)	**7.1763e+02** (**1.2643e+02**)
f_{19}	1.038e+03 (1.717e+00)	8.165e+02 (3.274e+02)	7.673e+02 (3.992e+02)	6.834e+02 (4.062e+02)	1.020e+03 (3.027e+02)	9.55e+02 (1.58e+02)	**2.1405e+02** (**2.1873e+00**)
f_{20}	8.9918e+02 (2.17e+01)	5.0001e+02 (5.7831e−02)	4.1123e+02 (4.2936e+00)	1.0889e+03 (3.015e+01)	5.000e+02 (1.311e−04)	2.15e+02 (9.07e−01)	**2.1023e+02** (**1.9173e+00**)

Table 5
Wilcoxon's Rank-sum test results of 30-dimensional problems f_1–f_{20}.

Functions	DE/rand/1/bin	SaDE	JADE	DMS-PSO	CMA-ES	IPOP-CMA-ES
f_1	+	+	+	+	+	+
f_2	+	+	+	+	+	+
f_3	+	+	+	+	+	+
f_4	+	+	+	+	+	+
f_5	+	+	+	+	+	+
f_6	+	+	+	+	+	+
f_7	+	+	+	+	+	+
f_8	+	+	+	+	+	−
f_9	+	−	−	+	+	−
f_{10}	+	−	−	−	=	−
f_{11}	+	+	+	+	+	+
f_{12}	+	+	+	+	+	+
f_{13}	+	+	+	+	+	+
f_{14}	+	+	+	+	+	+
f_{15}	+	+	+	+	+	+
f_{16}	+	+	+	+	+	+
f_{17}	+	+	+	+	+	+
f_{18}	+	+	+	+	+	=
f_{19}	+	+	+	+	+	+
f_{20}	+	+	+	+	+	+

Table 6
Wilcoxon's Rank-sum test results of 50-dimensional problems f_1–f_{20}.

Functions	DE/rand/1/bin	SaDE	JADE	DMS-PSO	CMA-ES	IPOP-CMA-ES
f_1	+	+	+	+	+	+
f_2	+	+	+	+	+	+
f_3	+	+	+	+	+	+
f_4	+	+	+	+	+	+
f_5	+	+	+	+	+	+
f_6	+	+	+	+	+	+
f_7	+	+	+	+	+	+
f_8	+	+	+	+	+	–
f_9	+	–	–	–	–	–
f_{10}	+	–	–	–	–	–
f_{11}	+	+	+	+	+	+
f_{12}	+	+	+	+	+	+
f_{13}	+	+	+	+	+	+
f_{14}	+	+	+	+	+	+
f_{15}	+	+	+	+	+	+
f_{16}	+	+	+	+	+	+
f_{17}	+	+	+	+	+	+
f_{18}	+	+	+	+	+	+
f_{19}	+	+	+	+	+	+
f_{20}	+	+	+	+	+	+

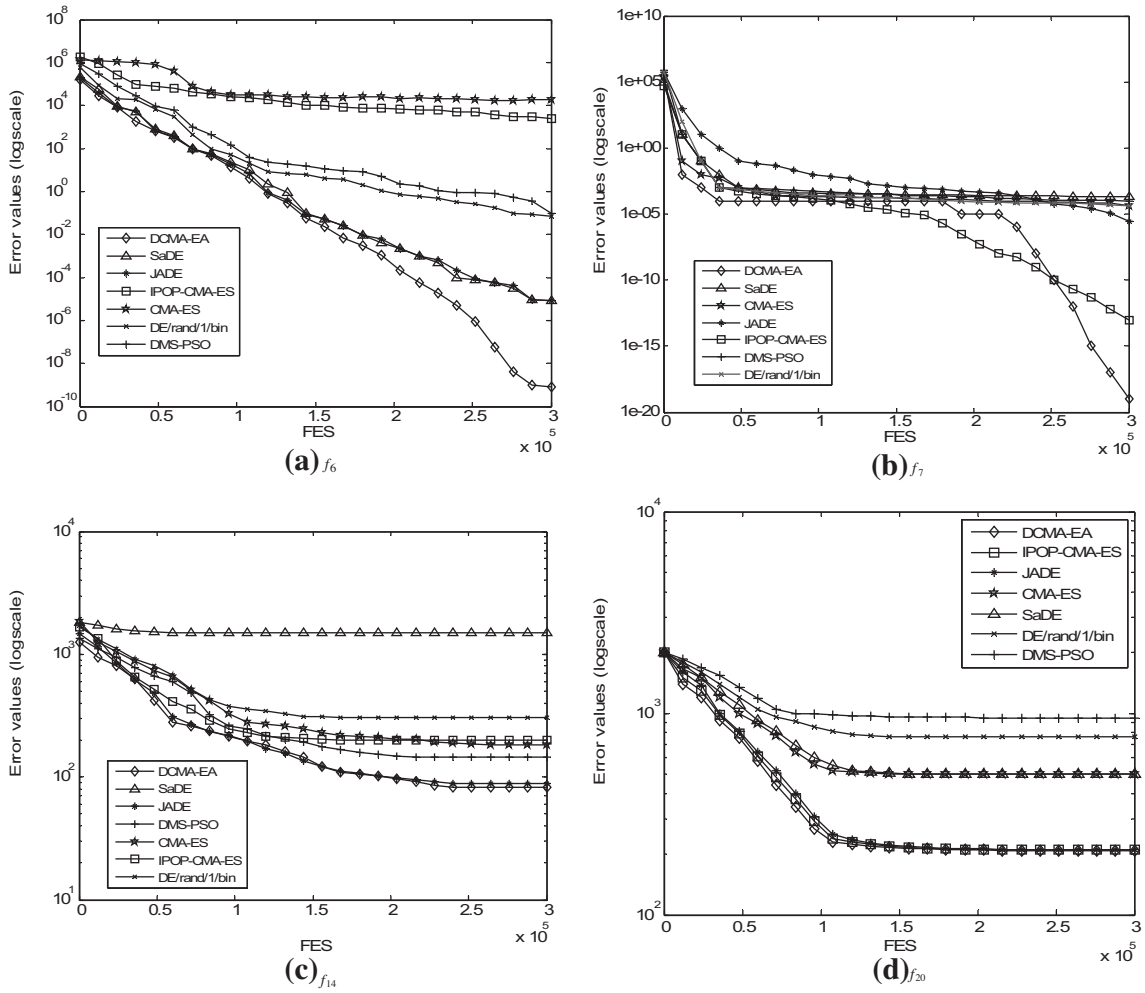

Fig. 2. Progress towards the optimum solution for median run of seven algorithms over four unconstrained test functions (30 D).

Table 7
Mean and standard deviation of the best-of-run solutions for 50 independent runs on f_6–f_9, f_{13}, f_{14}.and f_{19} in 30 dimensions. (Best entries are marked in bold.)

Functions	DCMA-EA with $Cr_m = 0.1$	DCMA-EA with F constant at 0.9	DCMA-EA with P decreasing from 0.5 to 0	DCMA-EA With $Cr_m = 0.9$, randomized F and the usual variation of P
f_6	2.465e+00 (1.984e+00)	6.273e−01 (3.827e−01)	5.473e−01 (2.738e−01)	**6.3702e−09** (**7.1254e−09**)
f_7	4.576e−02 (2.837e−02)	1.078e−02 (9.287e−03)	3.556e−03 (6.475e−03)	**1.1102e−20** (**1.1845e−20**)
f_8	2.105e+01 (7.836e−02)	2.092e+01 (4.494e−02)	2.096e+01 (3.647e−02)	**2.0903e+01** (**5.4523e−02**)
f_9	**1.4890e+01** (**9.4523e+00**)	3.995e+01 (8.364e+00)	2.975e+01 (8.274e+00)	5.0890e+01 (1.4523e+01)
f_{13}	2.364e+02 (4.375e+01)	1.423e+02 (3.475e+01)	1.253e+02 (1.364e+01)	**9.2307e+01** (**3.8872e+01**)
f_{14}	3.034e+02 (4.283e+01)	2.534e+02 (3.756e+01)	1.364e+02 (2.453e+01)	**9.9328e+01** (**3.9757e+01**)
f_{19}	2.125e+02 (2.293e+00)	2.115e+02 (3.289e+00)	2.103e+02 (1.024e+00)	**2.0986e+02** (**1.6271e+00**)

These parameters are chosen in accordance with the original literatures of CMA-ES [14–18]. The parameters that control the performance of DCMA-EA are the scale factor F, mean Cr_m of the crossover probability distribution and the control parameter P that controls the participation of the current generation mean vector and the corresponding target vector in generating the mutated vectors. To explain the rationale on the choice of values of these control parameters we consider three variants of DCMA-EA:

- DCMA-EA with F kept at a constant value (0.9) for all the particles.
- DCMA-EA with control parameter P being dynamically decreased from 0.5 to 0 with generations.
- DCMA-EA with Cr_m kept at a lower value (0.1) for all the functions.

We have compared these three variants of DCMA-EA with the original DCMA-EA algorithm on seven functions f_6, f_7, f_8, f_9, f_{13}, f_{14} and f_{19} chosen from the test functions described in Table 2. Among these functions f_9 is separable while rest are non-separable functions. Table 7 below represents the mean and the standard deviation (within parentheses) of the best-of-run errors for 50 independent runs of DCMA-EA and all the three variants of MDE_pBX considered in this section for the seven test functions mentioned above in 30 dimensions.

From the above results it is clear that if any of the three parameters F, Cr_m, and P is chosen in an altered manner then the performance of DCMA-EA deteriorates signifying the proper choice of values of these parameters. When the control parameter P is dynamically decreased from 0.5 to 0, the share of the current generation mean vector in generating the mutated vectors gradually increases with generations as evident from Eq. (37). So the population diversity reduces as all the particles get perturbed about the same mean vector and the population tends to become stagnant with generations weakening the convergence performance. In case of the scale factor F being kept at a constant value (0.9) all the vectors suffer perturbation by the same difference vector component reducing the population diversity to a large extent. On the other hand, low value of Cr_m (0.1) hampers the performance of DCMA-EA on non-separable functions while in case of the separable function f_9 the performance of DCMA-EA is enhanced for $Cr_m = 0.1$. Since most of the optimization problems are non-separable in nature, we prefer keeping Cr_m at a higher value (0.9) as it favours non-separable functions.

6. Application to real world optimization problems

Recently, various soft computing techniques have been applied to complex real-life problems arising from different fields in order to obtain superior results, e.g. see [5,28,36,50,51,55,58]. In this section we outline two famous real world optimization problems in order to judge the efficacy of the proposed hybrid algorithm DCMA-EA.

6.1. Application to spread spectrum radar poly-phase code design

A famous problem of optimal design arises in the field of spread spectrum radar poly-phase codes [9,33]. Such a problem is very well-suited for the application of global optimization algorithms like DE. The problem can be formally stated as:

$$\text{Global min } f(\vec{X}) = \max\{\varphi_1(\vec{X}), \ldots, \varphi_{2m}(\vec{X})\}, \tag{41}$$

where $\vec{X} = \{(x_1, \ldots, x_D) \in \Re^D | 0 \leqslant x_j \leqslant 2\pi, j = 1, \ldots, D\}$ and $m = 2D - 1$

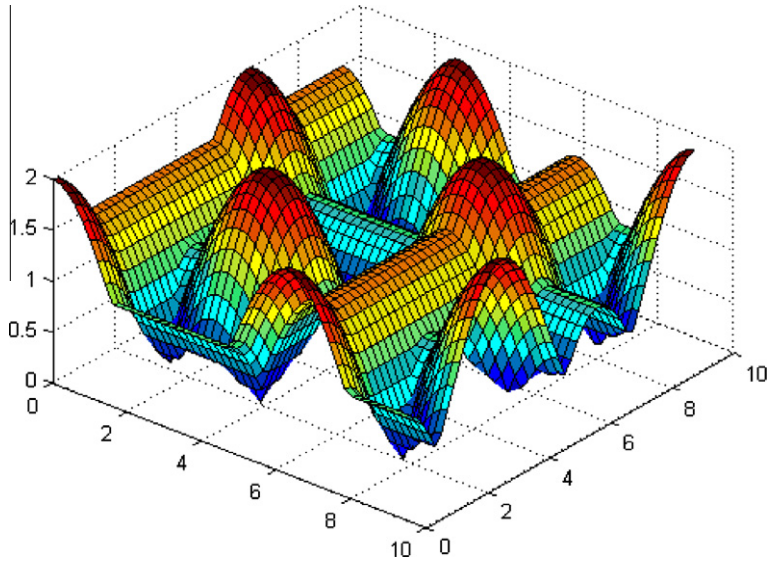

Fig. 3. Objective function landscape of (41) for $D = 2$.

Table 8
Average and standard deviation (in parentheses) of the best-of-run solutions for 50 independent runs over the spread-spectrum radar poly-phase code design problem (number of dimensions $D = 19$ and $D = 20$).

D	Mean best-of-run solution (std. dev.)							Statistical significance
	DE/rand/1/bin	JADE	SaDE	DMS-PSO	CMA-ES	IPOP-CMA-ES	DCMA-EA	
19	2.2071e+00 (5.0303e−02)	1.5183e+00 (8.5205e−02)	1.3592e+00 (4.6602e−02)	1.4056e+00 (3.7283e−02)	1.1345e+00 (3.8721e−01)	6.021e−01 (9.812e−02)	**5.0000e−01** (**2.2305e−03**)	+
20	3.2751e+00 (1.5944e−01)	2.3908e+00 (9.7657e−02)	1.8243e+00 (8.4491e−02)	1.9535e+00 (8.7911e−02)	1.5988e+00 (2.9843e−01)	6.176e−01 (1.232e−03)	**5.3130e−01** (**3.5767e−03**)	+

$$\varphi_{2i-1}(\vec{X}) = \sum_{j=i}^{D} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), \quad i = 1, 2, \ldots, D, \tag{42a}$$

$$\varphi_{2i}(\vec{X}) = 0.5 + \sum_{j=i+1}^{D} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k\right), \quad i = 1, 2, \ldots, D-1, \tag{42b}$$

$$\varphi_{m+i}(\vec{X}) = -\varphi_i(\vec{X}), \quad i = 1, 2, \ldots, m. \tag{42c}$$

According to [9] the above problem has no polynomial time solution. The objective function for $D = 2$ is shown in Fig. 3. Each variable is randomly initialized in the interval $[0, 2\pi]$. The search was kept confined in this region.

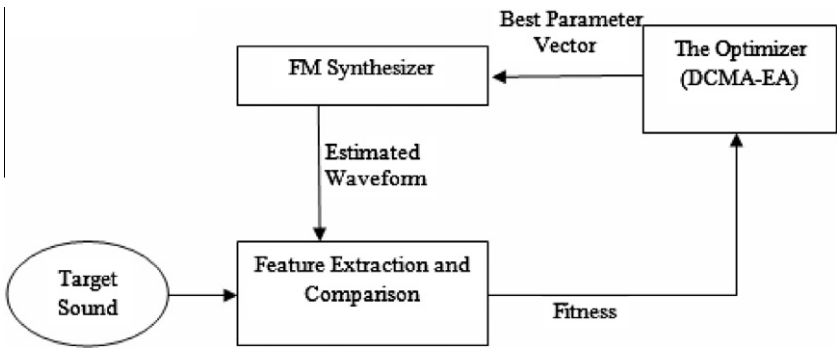

Fig. 4. Architecture of the optimization system.

Table 9
Average and standard deviation (in parentheses) of best-of-run solutions for 50 independent runs of seven algorithms on the Frequency Modulator Synthesis problem. Each algorithm was run for 10^5 FEs.

Mean best-of-run solution (std. dev.)							Statistical significance
DE/rand/1/bin	JADE	SaDE	DMS-PSO	CMA-ES	IPOP-CMA-ES	DCMA-EA	
1.7484e−01 (9.2680e−02)	1.8255e−08 (2.1585e−08)	7.8354e−08 (9.1254e−08)	1.4056e−06 (3.2765e−06)	8.4569e−06 (9.9721e−06)	9.5559e−09 (1.0356e−08)	**6.9865e−9** (**8.3667e−09**)	+

Table 8 provides the mean best-of-run objective function values and standard deviations obtained with six competitor algorithms as mentioned in section 5.2 over 50 independent runs. We consider two difficult instantiations of the code design problem with $D = 19$ and $D = 20$. For all the algorithms, each run was continued up to 3×10^5 FEs. The results indicate that the proposed DCMA-EA could beat the other six algorithms in a statistically significant manner for both the design instances.

6.2. Application to parameter estimation for frequency-modulated (FM) sound waves

Frequency-modulated (FM) sound synthesis plays an important role in several modern music systems. This section describes an interesting application of the proposed DCMA-EA algorithm to the optimization of parameters of an FM synthesizer. A few related works that attempt to estimate parameters of the FM synthesizer using the genetic algorithm can be found in [21,22]. Here we introduce a system that can automatically generate sounds similar to the target sounds. It consists of an FM synthesizer, a DE optimizer, and a feature extractor. The system architecture is shown in Fig. 4. The target sound is a *.wav* file. The algorithm initializes a set of parameters and the FM synthesizer generates the corresponding sounds. In the feature extraction step, the dissimilarities of features between the target sound and synthesized sound are used to compute the fitness value. The process continues until synthesized sounds become very similar to the target. The specific instance of the problem discussed here involves determination of six real parameters. $\vec{X} = \{a1, \omega1, a2, \omega2, a3, \omega3\}$ of the FM sound wave given by Eq. (43) for approximating it to the sound wave given in Eq. (44) where $\theta = 2\pi/100$. The parameters are defined in the range $[-6.4, +6.35]$. The formula for the estimated sound wave and the target sound wave may be given as:

$$y(t) = a1 \cdot \sin(\omega1 \cdot t \cdot \theta + a2 \cdot \sin(\omega2 \cdot t \cdot \theta + a3 \sin(\omega3 \cdot t \cdot \theta))) \tag{43}$$

$$y_0(t) = 1.0 \cdot \sin(5.0 \cdot t \cdot \theta - 1.5 \cdot \sin(4.8 \cdot t \cdot \theta + 2.0 \sin(4.9 \cdot t \cdot \theta))). \tag{44}$$

The goal is to minimize the sum of squared errors between the estimated sound and the target sound, as given by Eq. (45). This problem involves a highly complex multimodal function having strong epitasis (interrelation among the variables), with optimum value 0.0.

$$f(\vec{X}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2. \tag{45}$$

Owing to the great difficulty of solving this problem with high accuracy without specific operators for continuous optimization (like gradual GAs [21]), we stop the algorithm when the number of Function Evaluations exceed 10^5.

Table 9 indicates that DCMA-EA could achieve statistically superior performance over the six contestant algorithms as mentioned in section 5.2 for the Frequency Modulator Synthesis problem.

7. Conclusions

We propose a hybrid EA by combining the difference vector based mutation operator of DE into the main structure of the CMA-ES algorithm for solving real-parameter and single-objective optimization problems. The synergy of DE and CMA-ES is made in such a way, so as not to impose any computational burden on the main algorithm in terms of number of FEs. Combined with the crossover and the selection procedure of DE the hybrid algorithm can achieve a better trade-off between its explorative and exploitative tendencies.

We undertook a comparative study of DCMA-ES with classical CMA-ES and DE (DE/rand/1/bin), two state-of-the-art variants of DE (SaDE and JADE), a renowned variant of CMA-ES (IPOP-CMA-ES) and a competitive real optimizer of current interest (DMS-PSO) over a test-suite comprising 20 well-known numerical benchmarks along with two practical optimization problems. Our experimental results indicate that while DCMA-EA remains always better than its two ancestors CMA-ES and DE, it is also able to outperform the other state-of-the-art variants of the ancestors (SaDE, JADE and IPOP-CMA-ES) over most of the tested problem instances in a statistically meaningful way.

The presented work can be extended in multiple directions. There is much scope for enhancing the performance of DCMA-EA further. Future research may focus on incorporating the operators of recently developed DE variants like SaDE, JADE into CMA-ES as the mutation and crossover operators of these DE variants are much improved than those of DE/rand/1/bin. Further, more than one difference vectors may be incorporated into the mutation scheme of DCMA-EA and their effectiveness can be tested on critical landscapes. DCMA-EA should also be adopted in future for handling multi-objective and constrained optimization problems.

References

[1] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2005, pp. 1769–1776.
[2] T. Bäck, D. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Part D, Oxford Univ. Press, 1997.
[3] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks, Soft Computing – A Fusion of Foundations, Methodologies and Applications 13 (8) (2009) 811–831.
[4] X.S. Chen, Y.S. Ong, M.H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Transactions on Evolutionary Computation (2011), in press.
[5] C.T. Cheng, C.P. Ou, K.W. Chau, Combining a fuzzy optimal model with a genetic algorithm to solve multiobjective rainfall-runoff model calibration, Journal of Hydrology 268 (1–4) (2002) 72–86.
[6] S. Das, P.N. Suganthan, Differential evolution – a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[7] S. Das, A. Biswas, A. Abraham, On stability and convergence of the population-dynamics in differential evolution, AI Communications 22 (1) (2009) 1–20.
[8] R. Dawkins, The Selfish Gene, Oxford University Press, New York, 1976.
[9] M.L. Dukić, Z.S. Dobrosavljević, A method of a spread spectrum radar polyphase code design, IEEE Journal on Selected Areas in Communications 8 (1990) 743–749.
[10] S. Ghosh, S. Das, A.V. Vasilakos, K. Suresh, On convergence of differential evolution over a class of continuous functions with unique global optimum, IEEE Trans. SMC-B, 2011 (doi:10.1109/TSMCB.2011.2160625).
[11] C. Grosan, A. Abraham, Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews, Studies in Computational Intelligence (SCI), vol. 75, Springer, 2007. pp. 1–17.
[12] N. Hansen, The CMA evolution strategy: a comparing review, in: J.A. Lozano, P. Larrañga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms, Springer, 2006, pp. 75–102.
[13] N. Hansen, S. Kern, Evaluating the CMA evolution strategy on multimodal test functions, in: Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature PPSN VIII, Springer, Berlin, 2004, pp. 282–291.
[14] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, 1996, pp. 312–317.
[15] N. Hansen, A. Ostermeier, Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: the $(\mu/\mu_I, \lambda)$-ES, in: EUFIT'97, 5th Europian Congress on Intelligent Techniques and Soft Computing, Proceedings, Aachen, Verlag Mainz, Wissenschaftsverlag, 1997, pp. 650–654.
[16] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[17] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary Computation 11 (1) (2003) 1–18.
[18] N. Hansen, A. Ostermeier, A. Gawelczyk, On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation, in: L. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, Pittsburgh, Morgan Kaufman, 1995, pp. 57–64.
[19] W.E. Hart, N. Krasnogor, J.E. Smith (Eds.), Recent Advances in Memetic Algorithms, Studies in Fuziness and Soft Computing Series, vol. 166, Springer, 2005.
[20] W.E. Hart, N. Krasnogor, J.E. Smith, Memetic evolutionary algorithms, in: W.E. Hart, N. Krasnogor, J.E. Smith (Eds.), Recent Advances in Memetic Algorithms, Springer-Verlag, Berlin, Germany, 2004, pp. 3–27.
[21] F. Herrera, M. Lozano, Gradual distributed real-coded genetic algorithms, IEEE Transactions on Evolutionary Computation 4 (1) (2000) 43–62.
[22] A. Horner, J. Beauchamp, L. Haken, Genetic algorithms and their application to FM matching synthesis, Computer Music Journal 17 (4) (1993) 17–29.
[23] C. Igel, T. Suttorp, N. Hansen, A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategiesm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), ACM Press, 2006, pp. 453–460.
[24] Jastrebski, Arnold, Improving evolution strategies through active covariance matrix adaptation, in: Proceedings of the 2006 IEEE World Congress on Computational Intelligence, 2006, pp. 9719–9726.
[25] C. Li, S. Yang, T.T. Nguyen, E.L. Yu, X. Yao, Y. Jin, H.-G. Beyer, P.N. Suganthan, Benchmark generator for CEC'2009 competition on dynamic optimization, Technical Report, University of Leicester, University of Birmingham, Nanyang Technological University, September 2008.
[26] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of IEEE Swarm Intelligence Symposium (SIS) 2005, June, 2005.
[27] J.J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proceedings of the IEEE Swarm Intell. Symp., Pasadena, CA, June 2005, pp. 68–75.
[28] J.Y. Lin, C.T. Cheng, K.W. Chau, Using support vector machines for long-term discharge prediction, Hydrological Sciences Journal 51 (4) (2006) 599–612.
[29] R. Mallipeddi, P.N. Suganthan, Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies, in: Swarm Evolutionary and Memetic Computing Conference, Chennai, India, vol. 6466, 2010, pp. 71–78.
[30] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing 11 (2) (2011) 1679–1696, doi:10.1016/j.asoc.2010.04.024.
[31] A.C. Martínez-Estudillo, C. Hervás-Martínez, F.J. Martínez-Estudillo, N. García-Pedrajas, Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Transaction on System Man and Cybernatics B Cybern 36 (3) (2006) 534–545.
[32] S.K. Minhazul Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Transactions on Systems, Man and Cybernetics – Part B (2011), in press.
[33] N. Mladenović, J. Petrovic, V. Kovacevic-Vujicic, M. Cangalovic, Solving spread-spectrum radar polyphase code design problem by tabu search and variable neighborhood search, European Journal of Operational Research 153 (2003) 389–399.
[34] D. Molina, M. Lozano, C. García-Martínez, F. Herrera, Memetic algorithms for continuous optimization based on local search chains, Evolutionary Computation 18 (1) (2010) 27–63.
[35] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program (report 826), 1989.
[36] N. Muttil, K.W. Chau, Neural network and genetic programming for modelling coastal algal blooms, International Journal of Environment and Pollution 28 (3–4) (2006) 223–238.
[37] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Computational Intelligence Magazine 5 (2) (2010) 54–65.
[38] F. Neri, V. Tirronen, Recent Advances in Differential Evolution: A Review and Experimental Analysis, Artificial Intelligence Review, Springer 33(1) (2010) 61–106.
[39] F. Neri, V. Tirronen, Scale factor local search in differential evolution, Memetic Computing 1 (2) (2009) 153–171.
[40] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Information Sciences 181 (12) (2011) 2469–2487.
[41] Y.S. Ong, M.H. Lim, N. Zhu, K.W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man and Cybernetics – Part B 36 (1) (2006).
[42] P. Preux, E.-G. Talbi, Towards hybrid evolutionary algorithms, International Transactions in Operational Research 6 (6) (2006) 557–570.
[43] K. Price, R. Storn, J. Lampinen, Differential Evolution – A Practical Approach to Global Optimization, Springer, Berlin, 2005.

[44] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.
[45] A. Sinha, D.E. Goldberg, A survey of hybrid genetic and evolutionary algorithms, IlliGAL Report No. 2003004, Jan., 2003.
[46] R. Storn, K.V. Price, Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, ICSI, 1995. <http://www.http.icsi.berkeley.edu/~storn/litera.html>.
[47] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, Nanyang Technological University, Singapore, May 2005 and KanGAL Report #2005005, IIT Kanpur, India.
[48] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark functions for the CEC'2008 special session and competition on large scale global optimization, Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.
[49] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, T. Rossi, An enhanced memetic differential evolution in filter design for defect detection in paper production, Evolutionary Computation Journal 16 (4) (2008) 529–555.
[50] C.L. Wu, K.W. Chau, Y.S. Li, Predicting monthly streamflow using data-driven models coupled with data-preprocessing techniques, Water Resources Research 45 (2009) W08432, doi:10.1029/2007WR006737.
[51] J.X. Xie, C.T. Cheng, K.W. Chau, Y.Z. Pei, A hybrid adaptive time-delay neural network model for multi-step-ahead prediction of sunspot activity, International Journal of Environment and Pollution 28 (3–4) (2006) 364–381.
[52] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.
[53] D. Zaharie, On the explorative power of differential evolution, in: Proceedings of the Third International Workshop on Symbolic and Numerical Algorithms on Scientific Computing, SYNASC-2001, Timişoara, Romania, October 2–4, 2001.
[54] D. Zaharie, Statistical properties of differential evolution and related random search algorithms, in: P. Brito (Ed.), Proceedings of the International Conference on Computational Statistics, Porto, Portugal, August 24–29, Physica-Verlag, HD, 2008, pp. 473–485.
[55] J. Zhang, K.W. Chau, Multilayer ensemble pruning via novel multi-sub-swarm particle swarm optimization, Journal of Universal Computer Science 15 (4) (2009) 840–858.
[56] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.
[57] Q. Zhang, A. Zhou, S.Z. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition, Technical Report CES-887, University of Essex and Nanyang Technological University, 2008.
[58] J. Zhang, J. Zhuang, H. Du, S. Wang, Self-organizing genetic algorithm based tuning of PID controllers, Information Sciences 179 (7) (2009) 1007–1018.