Information Sciences 281 (2014) 443–461

Contents lists available at ScienceDirect

Information Sciences

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

# Optimal filter design using an improved artificial bee colony algorithm

Digbalay Bose [a], Subhodip Biswas [a], Athanasios V. Vasilakos [b,*], Sougata Laha [a]

[a] *Department of Electronics & Telecommunication Engineering, Jadavpur University, Kolkata 32, West Bengal, India*
[b] *Department of Computer Science, Kuwait University, Safat 3060, Kuwait*

A R T I C L E   I N F O

*Article history:*
Received 23 November 2013
Received in revised form 6 April 2014
Accepted 18 May 2014
Available online 4 June 2014

*Keywords:*
Analog filter design
Artificial bee colony algorithm
Global optimization
Information sharing
Passive circuit component
Swarm intelligence

A B S T R A C T

The domain of analog filter design revolves around the selection of proper values of the circuit components from a possible set of values manufactured keeping in mind the associated cost overhead. Normal design procedures result in a set of values for the discrete components that do not match with the preferred set of values. This results in the selection of approximated values that cause error in the associated design process. An optimal solution to the design problem would include selection of the best possible set of components from the numerous possible combinations. The search procedure for such an optimal solution necessitates the usage of Evolutionary Computation (EC) as a potential tool for determining the best possible set of circuit components. Recently algorithms based on Swarm Intelligence (SI) have gained prominence due to the underlying focus on collective intelligent behavior. In this paper a novel hybrid variant of a swarm-based metaheuristics called Artificial Bee Colony (ABC) algorithm is proposed and shall be referred to as CRbABC_*Dt* (Collective Resource-based ABC with Decentralized tasking) and it incorporates the idea of decentralization of attraction from super-fit members along with neighborhood information and wider exploration of search space. Two separate filter design instances have been tested using CRbABC_*Dt* algorithm and the results obtained are compared with several competitive state-of-the-art optimizing algorithms. All the components considered in the design are selected from standard series and the resulting deviation from the idealized design procedure has been investigated. Additional empirical experimentation has also been included based on the benchmarking problems proposed for the CEC 2013 Special Session & Competition on Real-Parameter Single Objective Optimization.

© 2014 Elsevier Inc. All rights reserved.

## 1. Introduction

The effective choice of circuit components in the domain of analog filter design is of paramount importance. Traditional designs include choosing the components to be equal to each other and this result in significant simplification of the design process. Nevertheless, the components like capacitors or resistors are available in terms of multiples of certain series like E12, E24, E48, E96 and E192, with their respective range of tolerances and component values. Keeping in mind the associated

* Corresponding author. Tel.: +965 2108074068.
*E-mail addresses:* digbose92@gmail.com (D. Bose), sub17was@gmail.com (S. Biswas), th.vasilako@gmail.com (A.V. Vasilakos), so15laha@gmail.com (S. Laha).

http://dx.doi.org/10.1016/j.ins.2014.05.033
0020-0255/© 2014 Elsevier Inc. All rights reserved.

cost constraints and the need for an efficient design, the resistor and capacitor values are chosen from these series. Since the values of the components are selected from a set of finite values associated with each series, a thorough search must be performed so that the resulting deviation in the search process is minimized. The search for an optimal set of component values from the numerous possible combinations appears to be an exhaustive process, thus diverting the attention towards the application of those heuristics which combines high efficiency with minimal computational time.

The necessity for an efficient search process has led practitioners to rely on Evolutionary Algorithms (EAs) [4], which mimic the process of natural evolution by simulating operators like mutation, crossover, recombination etc. The application of EAs in the domain of filter design has now gained widespread cognizance among the researchers as a potential topic of exploration and many notable works have been performed in this regard. A new design method for two dimensional recursive filters was explored by Mastorakis et al. [26] using Genetic Algorithm (GA) [15] by formulating the design problem as a constrained optimization problem and explored the application of metaheuristics in the design procedure of Infinite Impulse Response (IIR) filters. The applicability of GA in the design process of state variable filter was explored by Horrocks and Spittle [16] with the resistor values being selected from the E12 series. Vural et al. [39] investigated the performance of some existing EAs in the domain of analog filter design, primarily Butterworth [40] and state variable filter [41] synthesis. Chang et al. [8] proposed a novel tree based representation along with genetic programming for passive filter design. Xu and Ding [46] introduced an adaptive immune GA that combined both GA with immune algorithm for the purpose of improving the diversity, convergence speed in order to obtain an efficient system of EA based circuit design. Wang et al. [42] tested the combination of both Particle Swarm Optimization (PSO) [22] and Finite Element Method and used in the design process of microwave filters, having arbitrary geometries. Karaboga and Cetinkaya [21] explored the possibility of application of ABC algorithm [17–19] in the design process of adaptive Finite Impulse Response (FIR) and IIR filters. Das and Vemuri's [13] circuit synthesis framework for the passive analog circuits, based on a GA, was devised for obtaining the topology and the component values simultaneously. Sheta [36] used Differential Evolution (DE) [12,37,29] algorithm to select the elements of passive and active filter structures. Das and Konar [10,11] made use of modern swarm intelligent metaheuristics for 2-D IIR filter design.

This paper primarily investigates the applicability of a novel variant of the ABC algorithm called CRbABC_*Dt*(Collective Resource-base ABC with Decentralized tasking), which utilizes an artificial adaptive communication mechanism and a decentralized, directed exploration via neighborhood based dynamics towards the obtaining optimal solution. Our proposed algorithm has been tested on two design problems of a fourth order VCVS Butterworth filter and a second order state variable filter (SVF). The design problems have been formulated by considering the components (capacitors, resistors) from specified series. The results of our proposed algorithm have been compared with 6 existing algorithms which have reported competitive performance for global optimization.

The rest of the paper is organized as follows. Section 2 details the analog filter structures followed by the conventional design procedure in Section 3. A vivid description of the design problem formulation is provided in Section 4. Subsequent sections include elucidation of the classical ABC algorithm in Section 5 and its improved variant is proposed in Section 6. The associated experimental design and results are analyzed in Section 7 along with comparison on standard CEC '13 benchmark. Finally Section 8 concludes the paper.

## 2. Low pass analog active filter structures

Analog filters [32,35] are primarily devices that allow passage of electrical signals within a particular frequency range or at a particular frequency by restricting the passage of others. These filters are considered as the building blocks of many communication system, where they are used in wide range of applications in varying ranges of frequencies. Applications are manifold including speech processing, channel selection, demodulation of signals, separating the signal from the noise, etc. Active analog filters utilize op-amps as the active component, which are combined with resistors and capacitors in certain arrangements to obtain desired filter response characteristics.

Active analog filters enjoy a significant advantage over the passive design due to their ability to be packed in a miniaturized format owing to efficient IC design. In spite of this, several limitations exist in the form of slew rate, unwanted noise, gain bandwidth product, etc.

The types of filter generally include the given categories- high pass, low pass, band pass and band reject.

- *Passband gain*: Pass band gain is defined in the frequency range of interest and is ideally considered to be unity or at a fixed value.
- *Cutoff frequency*: Cutoff frequency determines the frequency at which the response makes a transition between pass band and stop band.
- *Quality factor*: Quality factor determines the sharpness of the response curve.

An ideal low pass filter is characterized by a uniform pass band up to a certain frequency called *cut-off frequency* after which the higher order frequency components are attenuated. Ideal low pass filter characteristics include sharp distinction between the pass band and stop band as shown in the Fig. 1. The two low pass filter designs considered in this paper, namely state variable and Butterworth, are detailed below.

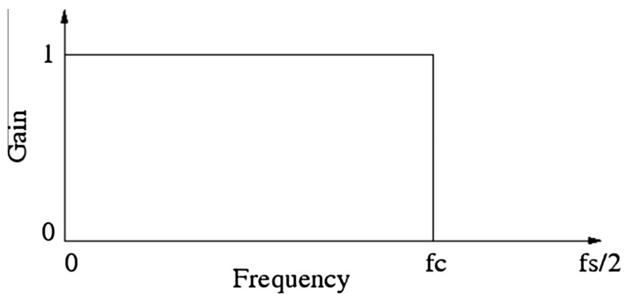


**Fig. 1.** Idealized low pass filter characteristics.

### 2.1. Butterworth filter

Butterworth filters are primarily characterized by the maximal flatness in its response plot in the pass band. The maximal flatness of the response plot of an $N^{th}$ order filter arises due to the disappearance of the $(2N-1)$ derivatives [32] of the magnitude of the transfer function of the filter near $\omega = 0$ i.e. at frequencies less than the cutoff frequency. The increase of the flatness of the pass band occurs with an increase in the order of the filter, where the pass band for a low pass filter ranges from $\omega = 0$ to $\omega = \omega_{cutoff}$ ($\omega_{cutoff}$ is the cutoff frequency) and the stop band lies in the range $\omega = \omega_{cutoff}$ to $\omega = \infty$. The design topology of the Butterworth filter is same as that considered in [39], where a fourth order Butterworth filter is designed by cascading two 2nd order Butterworth filter sections, such that the overall transfer function is given by the product of the transfer functions of the two sections. If $H_1(s)$ and $H_2(s)$ are the transfer functions of the first and second filter sections then the overall transfer function $H(s)$ is given as

$$H(s) = H_1(s) \cdot H_2(s) \tag{1}$$

If $\omega_{cutoff1}$ is the cutoff frequency of the first 2nd order low pass Butterworth segment and $Q_{b1}$ be its corresponding quality factor then $H_1(s)$ is

$$H_1(s) = \frac{\omega^2_{cutoff1}}{s^2 + s\frac{\omega_{cutoff1}}{Q_{b1}} + \omega^2_{cutoff1}} \tag{2}$$

Similarly, if $\omega_{cutoff2}$ is the cutoff frequency of the second filter segment and $Q_{b2}$ is the corresponding quality factor then $H_2(s)$ is

$$H_2(s) = \frac{\omega^2_{cutoff2}}{s^2 + s\frac{\omega_{cutoff2}}{Q_{b2}} + \omega^2_{cutoff2}} \tag{3}$$

The overall transfer function $H(s)$ of the fourth order low pass Butterworth filter, obtained by cascading the above two segments, is as obtained below

$$H(s) = \frac{\omega^2_{cutoff1}}{s^2 + s\frac{\omega_{cutoff1}}{Q_{b1}} + \omega^2_{cutoff1}} \cdot \frac{\omega^2_{cutoff2}}{s^2 + s\frac{\omega_{cutoff2}}{Q_{b2}} + \omega^2_{cutoff2}} \tag{4}$$

The transfer function of the fourth order low pass Butterworth filter, as shown in Fig. 2, is given below in terms of the discrete circuit components.

$$H(s) = \frac{1}{s^2 \cdot (R_a \cdot R_b \cdot C_c \cdot C_d) + s \cdot (R_a \cdot C_a + R_b \cdot C_a) + 1} \cdot \frac{1}{s^2 \cdot (R_c \cdot R_d \cdot C_a \cdot C_b) + s \cdot (R_c \cdot C_c + R_d \cdot C_c) + 1} \tag{5}$$

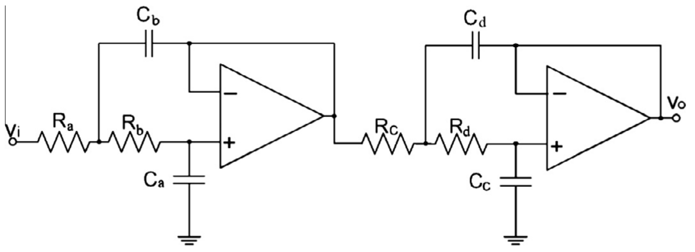


**Fig. 2.** Design of fourth order Butterworth low pass filter.

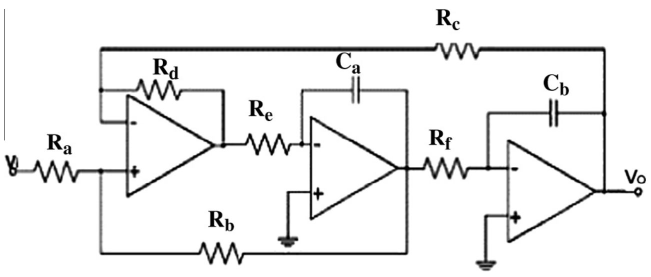


**Fig. 3.** Design of state variable low pass filter.

Comparing Eqs. (3)–(5), the expressions for cut-off frequency and quality factors are obtained as

$$\omega_{cutoff1} = \frac{1}{\sqrt{R_a \cdot R_b \cdot C_a \cdot C_b}} \tag{6}$$

$$\omega_{cutoff2} = \frac{1}{\sqrt{R_c \cdot R_d \cdot C_c \cdot C_d}} \tag{7}$$

$$Q_{b1} = \frac{\sqrt{R_a \cdot R_b \cdot C_a \cdot C_b}}{R_a \cdot C_a + R_b \cdot C_a} \tag{8}$$

$$Q_{b2} = \frac{\sqrt{R_c \cdot R_d \cdot C_c \cdot C_d}}{R_c \cdot C_c + R_d \cdot C_c} \tag{9}$$

## 2.2. State variable filter

The second filter design considered is that of the state variable filter, which is the most versatile among all the possible designs of active filters [36]. State variable design permits the simultaneous implementation of low pass, band pass and high pass filters in the same circuit and even allows the design of band reject circuits with the addition of an adder circuit.

The SVF design is inspired by the state space model of a linear time invariant system and it implements the state space model completely. The design involves interconnections between integrators and an adder circuit, where the constituent components are composed of resistors and capacitors. The $n^{th}$ order SVF in general consists of $n + 2$ OPAMPS ($n$ OPAMPS correspond to $n$ integrator circuits and two other OPAMPS are used for adder and output simulation). The output voltage of each integrator is associated with one of the state variables of the state space model. The associated SVF design is represented in Fig. 3 and the corresponding low pass response is obtained from the rightmost integrator output.

The response of the second order SVF is specified mainly by cut-off frequency $\omega_{svf}$, pass band gain $Hpass_{svf}$ and quality factor $Q_{factor\ \mathrm{svf}}$. These quantities are determined in terms of the passive components i.e. resistor and capacitor values using the following equation:

$$Hpass_{svf} = \frac{R_b}{R_c} \cdot \left(\frac{R_c + R_d}{R_a + R_b}\right) \tag{10}$$

$$\omega_{svf} = \sqrt{\frac{R_d}{R_c} \cdot \left(\frac{1}{C_a \cdot C_b \cdot R_e \cdot R_f}\right)} \tag{11}$$

$$Q_{factor\ svf} = \frac{R_c}{R_a} \cdot \left(\frac{R_a + R_b}{R_c + R_d}\right) \sqrt{\frac{C_a \cdot R_d \cdot R_e}{C_b \cdot R_c \cdot R_f}} \tag{12}$$

The values chosen for $\omega_{svf}$ and $Q_{factor\ \mathrm{svf}}$ are 10 k rad/s and 0.707 respectively. The $Hpass_{svf}$ is in general fixed at some particular value in case of traditional design methodologies, but the pass band gain can take any value from the continuous real space while EAs are applied.

## 3. Conventional design procedures

Conventional procedures involve setting all the resistors equal to a particular value of resistance. For Butterworth filter design problem, cut-off frequencies $\omega_{cutoff1}$ and $\omega_{cutoff2}$ are set to 10 k rad/s and the quality factors $Q_{b1}$ and $Q_{b2}$ are set to 1.3065 and 0.5412 respectively. Similar methods are applied for SVF design with all the resistor values set to a fixed value, and the cut-off frequency and the quality factors set to 10 k rad/s and 0.707 respectively. The capacitor values for both Butterworth and state variable design are then determined using Eqs. (6)–(12) respectively. In order to ensure that the component values lie within a feasible range, the resistor values are multiplied by a multiplying factor, called *magnitude scaling factor*, which is used to divide the capacitor values. For designing the filter according to values specified by the manufacturer's series, the exact values obtained are rounded off to any particular value within the manufacturer's series. Since the exact values are not considered for design due to constraint imposed by manufacturer's specifications, the associated design error increases. The method of conventional design is illustrated below for both Butterworth and state variable design problem.

Considering the transfer function for the first segment of the 4th order Butterworth filter structure, as given by (2), the conventional procedure is initiated by equating the values of two resistors $R_a$ and $R_b$ equal to a common value $R$ being $1\,\Omega$ and considering distinct values for both the capacitors $C_a$ and $C_b$. The cutoff frequency initially, $\omega_{cutoff1}$ is considered to be 1 rad/s and $Q_{b1}$ fixed at 1.3065. Then from Eqs. (2), (6) and (8), the capacitor values $C_a$ and $C_b$ are calculated as:

$$2/C_a = 0.765 \tag{13}$$

$$C_a \cdot C_b = 1 \tag{14}$$

In order to obtain a desired cutoff frequency value, $\omega_{cutoff1}$ (desired) for the low pass Butterworth filter segment, a frequency scaling factor *fscale* is determined as follows:

$$fscale = \frac{\omega_{cutoff1}(desired)(in\ \mathrm{rad/s})}{1\ \mathrm{rad/s}} \tag{15}$$

For obtaining practically realizable values for the resistor components, the resistor $R$, having resistance of $1\Omega$, is multiplied by a magnitude scaling factor called *mscale*. For example, to obtain $1k\Omega$ resistance, the *mscale* value is set at *1000*. Utilizing both the magnitude and frequency scaling factors, *mscale* and *fscale*, the resulting values of the capacitors are then obtained keeping in mind the cutoff frequency requirement as follows:

$$C'_a = \frac{C_a}{mscale \times fscale} \tag{16}$$

$$C'_b = \frac{C_b}{mscale \times fscale} \tag{17}$$

Similar methodology is followed in the conventional design procedure of the SVF with the values of all the resistors except $R_b$, set to a $R$, and $C_a$ and $C_b$ are chosen equal to $C$. Eq. (11) reduces to

$$R \cdot C \cdot \omega_{svf} = 1 \tag{18}$$

The value of the resistor $R_b$ is then obtained using (12) as:

$$R_b = (2 \cdot Q_{factor\ svf} - 1) \cdot R \tag{19}$$

The pass band gain value $Hpass_{svf}$, thus obtained is equal to 1 and a pair of values for $R$ and $C$ is chosen to satisfy (18). Generally, for state variable design process, a particular preferred resistor value is chosen for $R$, resulting in equal values for $R_a$, $R_c$, $R_d$, $R_e$ and $R_f$. The values of $C_a$, $C_b$ and $R_b$ are obtained using (18) and (19) respectively.

## 4. Design problem

EAs can be applied in filter design, particularly in the case of above mentioned filter structures, by developing the design problem in the form of a cost function by taking into consideration all the associated design parameters and constraints as well. Since the task is to obtain the best possible combination of the discrete circuit components, the design problem is considered to be of the form of a function which is to be minimized in order to obtain the required parameter values within a specified range.

The formulation of the problem considered in this paper is inspired from the work of Vural et al. [39], in which the cost function includes components to assess the accuracy in the measurements of both cut-off frequency and quality factor, so that the design error is minimized. The selection of the capacitors and resistors having required values is restricted within a certain range, i.e. $10^3 - 10^6\,\Omega$ for resistors and $10^{-9} - 10^{-6}$ F for capacitors.

### *4.1. Butterworth filter design problem*

Vural *et. al.* [39] investigated the effectiveness of GA, ABC and PSO in the particular design problems of Butterworth and state variable filters. In order to investigate the effectiveness of the proposed algorithm in comparison to other algorithms, the same form of the cost function is considered here. It takes into account the total error associated with the design process which includes both the error associated with the quality factor as well as the cut-off frequency.

If the cost function errors associated with cut-off frequency and quality factor are given as $CostF_\omega$ and $CostF_Q$ respectively, then the total error cost function associated with the design process as given in [35] is

$$CostF^{BW}_{error} = \frac{1}{2}(CostF_\omega + CostF_Q) \tag{20}$$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | $\alpha_1$ | $\beta$ | $\beta_1$ | $\gamma$ | $\gamma_1$ | $\delta$ | $\delta_1$ | $\varepsilon$ | $\varepsilon_1$ | $\theta$ | $\theta_1$ | $\vartheta$ | $\vartheta_1$ | $\rho$ | $\rho_1$ |

**Fig. 4.** Representation of the 16 dimensional solution vector.

The errors associated with cut-off frequency and quality factors are given as follows:

$$CostF_{\omega} = \frac{1}{\omega_c}(|\omega_{cutoff1} - \omega_c| + |\omega_{cutoff2} - \omega_c|) \tag{21}$$

$$CostF_{Q} = \left(\left|Q_{b1} - \frac{1}{0.7654}\right| + \left|Q_{b2} - \frac{1}{1.8478}\right|\right) \tag{22}$$

Here $\omega_{cutoff1}$, $\omega_{cutoff2}$, $Q_{b1}$ and $Q_{b2}$ are given by (6)–(9) respectively.

The value of $\omega_c$ here is fixed at 10 k rad/s. It is desired that the EA must minimize the total error cost function $CostF^{BW}_{error}$ so that a proper combination of the component values are obtained which would reduce the error cost function to a value approaching zero ideally.

### 4.1.1. Representation scheme for components in Butterworth filter design

A particular representation of the initial population is considered as in [39]. The initial population is taken to be a matrix having a size of $NP \times 16$ where $NP$ and 16 denote the population number and dimensionality of the solution vectors respectively. The representation of each 16$D$ trial solution vector is given as follows:

The elements of the solution vector are used for the determination of the appropriate values of the resistors and capacitors via a mapping scheme as detailed in Fig. 4:

$$\begin{aligned}
&R_a = \alpha \cdot 10^{\alpha_1+2} \quad (\Omega) \quad R_b = \beta.10^{\beta_1+2} \quad (\Omega)\\
&R_c = \gamma \cdot 10^{\gamma_1+2} \quad (\Omega) \quad R_d = \delta \cdot 10^{\delta_1+2} \quad (\Omega)\\
&C_a = \varepsilon \cdot 10^{\varepsilon_1+2} \quad (pF) \quad C_b = \theta \cdot 10^{\theta_1+2} \quad (pF)\\
&C_c = \vartheta \cdot 10^{\vartheta_1+2} \quad (pF) \quad C_d = \rho \cdot 10^{\rho_1+2} \quad (pF)
\end{aligned} \tag{23}$$

Additionally, certain design constraints have to be satisfied in order that the resistor values reside within the range $10^3 - 10^6\,\Omega$ (E12 value) and the capacitor values must lie within the specified E12 range of $10^{-9} - 10^{-12}$ F. The bounds associated with the design procedure are specified as:-

$$\begin{aligned}
&0.1 \leqslant \alpha, \beta, \gamma, \delta \leqslant 0.822 \quad 2 \leqslant \alpha_1, \beta_1, \gamma_1, \delta_1 \leqslant 4\\
&0.1 \leqslant \varepsilon, \theta, \vartheta, \rho \leqslant 0.822 \quad 2 \leqslant \varepsilon_1, \theta_1, \vartheta_1, \rho_1 \leqslant 4
\end{aligned} \tag{24}$$

## 4.2. State variable filter design problem

The cost function associated with the SVF design problem consists of two components pertaining to the errors in cutoff frequency and quality factor. If the cost function errors associated with cut-off frequency and quality factor are given as $CostF_{\omega}$ and $CostF_{Q}$ respectively, then the total error associated with the state variable design process is given as below:

$$CostF_{error\ svf} = \frac{1}{2}(CostF_{\omega} + CostF_{Q}) \tag{25}$$

$$CostF_{\omega} = \frac{1}{\omega_{CO}}|\omega_{svf} - \omega_{CO}| \tag{26}$$

$$CostF_{Q} = \frac{1}{Q_{CO}}|Q_{factor\ svf} - Q_{CO}| \tag{27}$$

In the above expressions equations $\omega_{svf}$ and $Q_{factor\ \mathrm{svf}}$ are given by (11) and (12) respectively. The values of $\omega_{CO}$ and $Q_{CO}$ are set to 10 k rad/s and 0.707 respectively. The EAs are operated on the cost function given by (25), such that a set of values of the discrete circuit components are obtained, which would reduce the cost function nearly to zero.

### 4.2.1. Representation scheme for components in state variable filter design

In the problem of state variable synthesis the representation of the initial population is identical to that considered in the case of Butterworth filter synthesis. The resistances and capacitances values are considered primarily from the standard E24 and E96 series [41]. A particular mapping scheme has been devised in order to determine the values of the capacitors and resistors from the elements of the solution vector. The mapping scheme is detailed as follows:-

$$\begin{aligned}
&R_a = \alpha \cdot 10^{\alpha_1+2} \quad (\Omega) \quad R_b = \beta \cdot 10^{\beta_1+2} \quad (\Omega)\\
&R_c = \gamma \cdot 10^{\gamma_1+2} \quad (\Omega) \quad R_d = \delta \cdot 10^{\delta_1+2} \quad (\Omega)\\
&R_e = \varepsilon \cdot 10^{\varepsilon_1+2} \quad (\Omega) \quad R_f = \theta.10^{\theta_1+2} \quad (\Omega)\\
&C_a = \vartheta \cdot 10^{\vartheta_1+2} \quad (pF) \quad C_b = \rho \cdot 10^{\rho_1+2} \quad (pF)
\end{aligned} \tag{28}$$

In the SVF synthesis problem the component values from the E24 standard series are first considered. In order that the resistors and capacitors must have an E24 value in the range $10^3 - 10^6\ \Omega$ and $10^{-9} - 10^{-12}$ F respectively, certain design limitations have to be satisfied. The bounds associated with the design procedure are

$$\begin{aligned} &0.1 \leqslant \alpha, \beta, \gamma, \delta \leqslant 0.91 \quad 2 \leqslant \alpha_1, \beta_1, \gamma_1, \delta_1 \leqslant 4 \\ &0.1 \leqslant \varepsilon, \theta, \vartheta, \rho \leqslant 0.91 \quad 2 \leqslant \varepsilon_1, \theta_1, \vartheta_1, \rho_1 \leqslant 4 \end{aligned} \tag{29}$$

In the second case of SVF design, component values from the E96 series were considered. In order that the components attain values compatible with the E96 series, the design constraints are detailed as below:

$$\begin{aligned} &0.1 \leqslant \alpha, \beta, \gamma, \delta \leqslant 0.976 \quad 2 \leqslant \alpha_1, \beta_1, \gamma_1, \delta_1 \leqslant 4 \\ &0.1 \leqslant \varepsilon, \theta, \vartheta, \rho \leqslant 0.976 \quad 2 \leqslant \varepsilon_1, \theta_1, \vartheta_1, \rho_1 \leqslant 4 \end{aligned} \tag{30}$$

## 5. Artificial bee colony algorithm

Bee colonies are hierarchical distributed systems just like other insect societies. Entomologists are of the view that in spite of the simplicity of individuals, an insect colony presents a highly structured social organization, by the virtue of which bee colonies can accomplish highly complex tasks surpassing the individual capabilities of a single bee. Drawing the analogy of their behavioral pattern and mapping them to the field of optimization led to the birth of a new branch of swarm meta-heuristics called Artificial Bee Colony algorithm, introduced by Karaboga [17] in 2005 for solving global optimization problems initially.

The entire bee-colony is in itself a large cluster of specialized individuals and the main focus is on the honeybee foragers entrusted with the duty of finding the food sources and collecting nectar from them. Let us consider that the food sources are distributed continuously throughout the solution space and each one has a nectar content associated with it. Each food source has an associated honey bee forager that continually evaluates its fitness value (nectar content). The position of a forager around a food source represent the probable solutions to the objective function that are gradually improved through the combined sequence of actions- positional perturbation, greedy selection, etc. Analogically the forager workforce represents the members of a population based metaheuristic. The steps involved in the ABC algorithm are as follows.

### 5.1. Initialization

The food sources indicate the trial solutions to the objective function. The search for an optimal solution in the real parameter space $R^n$ is initiated by random initialization of the trial solutions within certain pre-determined bounds and can be represented as:

$$\vec{\theta}_i^C = \left\{\theta_{i,1}^C, \theta_{i,2}^C, \ldots, \theta_{i,n}^C\right\} \tag{31}$$

where $n$ is the dimensionality of the search space, $C$ is the current cycle number, and $i$ denotes the food source associated with a forager and number that take values from 1 to $SN$ (Food Number). Each component of the food source is initialized using uniform random number as follows:

$$\theta_{i,j}^0 = \theta_j^{\min} + rand(0,1) \cdot \left(\theta_j^{\max} - \theta_j^{\min}\right) \tag{32}$$

The bounds of the search space are given by:-

$$\begin{aligned} \vec{\theta}_{\min} &= \left\{\theta_1^{\min}, \theta_2^{\min}, \ldots, \theta_n^{\min}\right\} \\ \vec{\theta}_{\max} &= \left\{\theta_1^{\max}, \theta_2^{\max}, \ldots, \theta_n^{\max}\right\} \end{aligned} \tag{33}$$

The termination criterion used here is satisfied when $C$ reaches the maximum cycle number ($MCN$).

### 5.2. Employed bee phase

Each food source has an employed forager associated with it and the population (colony) size is equal to the number of food sources. An employed bee modifies the position of the food source and searches the neighborhood of the employed food source. The original ABC algorithm relies on the association of a single forager with a particular food source, which then exploits the local (neighboring) food site by using the following equation

$$v_{i,j}^C = \theta_{i,j}^C + \varphi_{i,j} \cdot \left(\theta_{i,j}^C - \theta_{k,j}^C\right) \tag{34}$$

where $k \neq i$; $j \in \{1, 2, \ldots, n\}$; $i,k \in \{1, 2, \ldots, SN\}$ and $\varphi_{i,j}$ is a scaling factor randomly chosen between $[-1, 1]$. The perturbation is done for a single component $j$ so that local regions are explored. In case the modified value exceeds the search space bounds, it is set within the specified bounds.

Now the fitness of the newly produced site is calculated. The fitness function is calculated based on the objective (maximize or minimize). Here the objective is minimization and accordingly the fitness function is designed as

$$fitness_i = \left\{ \begin{array}{ll} 1/(1+f(\vec{\theta}_i)) & \text{if } f(\vec{\theta}_i) \geqslant 0 \\ 1+|f(\vec{\theta}_i)| & \text{if } f(\vec{\theta}_i) < 0 \end{array} \right\} \tag{35}$$

where $f(.)$ denote the objective function. Between $\vec{\theta}_i(c)$ and $\vec{v}_i(c)$, the fitter one is selected by the process of greedy selection [19]. In case the employed forager fails to improve upon the fitness of its assigned food source, the associated trial counter is incremented by 1 otherwise it is reset to 0.

### 5.3. Calculating probability

The employed foragers gather the information about the nectar content of the food sources and convey it to the waiting onlooker bees in the hive. The onlooker bees select the food source based on its nectar content. The chances of selecting a food source is determined by its probability value computed as

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{36}$$

There is a proportional rise in the onlooker population employed at a fitter food source.

### 5.4. Onlooker phase

This phase is identical to the employed bee phase except the fact that a uniform random number is generated in the range [0,1] and if this value is less than or equal to the probability (calculated above) of a given food source, positional modification using (34) is done. This is done till all the onlookers have been allotted a food source. Then an identical greedy selection mechanism is applied and trial counter values are accordingly updated. Note that in the original ABC algorithm the number of onlooker bees equaled the employed bee population size. Karaboga analyzed the performance of the ABC algorithm in detail in [19].

### 5.5. Scout phase

The scout phase was designed to prevent the wastage of computational resources. In case the trial counter for a food source exceeds a pre-defined *limit* then it is deemed to be exhausted such that no further improvement is possible. In such a case, it is randomly reinitialized using (32), its fitness value is calculated and the trial counter is reset to 0. This is motivated from the fact that repeated exploitation of nectar by a forager causes the exhaustion of the nectar present in the food source. The scout bees simulate the random walks performed by the decommissioned foragers [18,19] with the hope of accidental discovery of promising fitness basins.

### 5.6. Why ABC?

ABC algorithm requires only few control parameters to control its performance in relation to other EAs mostly. Although this simplicity comes at the cost of slow convergence rate and reduced exploration, adjustments to the framework can be made to ensure that ABC's performance improves in comparison to its classical form. Given below is a series of improvements done in order to ensure it is able to achieve a commendable performance in real time systems.

### 5.7. Need for improvement through hybridization

There has been a trend to hybridize algorithms to yield a novel variant as the outcome. Fundamentally, hybridization has been defined in [31] as "the mixing of at least two heterogeneous entities either through conscious manipulation or as a natural progressive transformation". From an algorithmic perspective, combining two or more distinct search methodologies in a synergistic manner can boost up the problem-solving ability of the hybridized variant. Thus hybridization is important in context of memetic computing techniques. Memetic algorithm (MA), coined in 1989 [27] as a form of hybrid global–local heuristic search methodology, is one of the foremost and rapidly growing areas of memetic computing research. Interested readers can refer to some well-known literary surveys [31,28,9] and the references therein to gain a better understanding.

The population-based optimization algorithms can be broadly grouped into EAs and SI-based algorithms, each with their unique advantages and disadvantages in the face of varying problem features and they very well complement each other. Thus there has been a trend to hybridize search methodologies such as EAs and Simulated Annealing [34], PSO, DE [45], etc. It has mostly resulted in novel variant with elated performance. With the advent of ABC, there were empirical investigations [1,3] that led to discovery of drawbacks in the search move like narrow search range due to single parameter perturbation, ignoring the fitness to reward ratio while selecting food sites, and absence of environmental factors in the algorithm design. Thus to improve the performance, there has been attempts to hybridize ABC with well-known search techniques as can be found in the survey by Karaboga [20].

Based on this observation, the authors came up with few logical improvement schemes based on the reported studies. Firstly, the need was felt to modify the basic perturbation equation and create a pool of perturbation modes and was implemented in one of our recent works [5] and idea have been applied here as well. This was done considering the diversity in the landscape type and problem features may be tackled well by an ensemble of perturbation modes. This is inspired from DE algorithm [12] which has well-known mutation (perturbation) strategies. Secondly, the lack of search range (due to single component perturbation) can be mitigated by using a multi-component perturbation employed by algorithms like DE and PSO. Thirdly, the most important aspect of our algorithm is the information sharing based on neighborhood members. This is needed to preserve diversity [30,38] as well as to exchange information from the super-fit members [7]. To ensure successful exploration by the search agents (foragers) without crowding, the primary focus has been on the T/E ratio (discussed later) while designing the algorithm outlined next. This has resulted in a two phase perturbation. The employed bee phase applies the pool of strategies to diversify the search logic while the onlooker bee phase ensures decentralization and thereby prevents overcrowding by carefully considering the T/E ratio. Lastly, the scaling (perturbation) factor $\varphi_{i,j}$ generating mechanism of the classical ABC has been modified. One might notice that algorithms like PSO and DE (/current-to-rand/1), which uses differential vector addition without mixing of components in search logic, suffers from high convergence tendency in multimodal fitness landscapes in spite of being rotationally invariant. To do away with it, Levy Distribution [23] due to its infinite second moment, is applied to create a greater degree of randomness in the modified food source positions generated by the foragers.

Having discussed the underlying motivation behind our algorithm design, working details are presented in the next section.

## 6. Our proposed approach: CR*b*ABC_*Dt* algorithm

The salient features of our algorithm following the initialization are detailed as below:

### *6.1. Multi-perturbation employed bee phase (wider search)*

The efficiency of a search method depends on its ability to promote diversification and local search simultaneously for efficient tracking of the optimum. In the basic variant of ABC, only one particular randomly selected parameter was selected for perturbation, resulting in a narrowed rectilinear locomotion over the search space and severely constricting the search range at the disposal of the population members.

A modification has been implemented here by the inclusion of the multi-perturbation employed bee phase, which is an improved modification of the employed bees phase and incorporates a sampled difference between pairs of food sources (after being scaled down by a suitable factor) which is added to the original food source via vector addition rather than the single component perturbation of classical ABC. In fact this procedure very well resembles the search equation in PSO [22] but this does not include the learning factors involved in PSO. Also the storage of memory the *pbest* location for each member (as in PSO) is relaxed here. These employed bees are responsible for executing locomotion in the search space by deriving information from their associated neighborhood. This modified phase can be illustrated by the following perturbation strategies based on the nomenclature proposed in [5]:

*a.* ABC—{*sto*}{1}: In this movement strategy the bee simultaneously spots three adjoining food sites and targets one among them based on fitness value as its source, and moves from it along the displacement vector obtained from the other two sites. It is an explorative-motion variant.
*b.* ABC—{*src*}{p-*fit*}{2}: A more exploitative-motion variant, this strategy utilizes information about the fittest found member from a sorted group comprising of *p*% of the total food sources and directs the employed bee from its source site along the direction pointing from the *p* fittest site towards a randomly selected site.
*c.* ABC—{*sto*}{p-*fit*}{2}: This locomotion strategy obtains a tradeoff between exploration and exploitation and is adept in overcoming local traps and reaching fitter regions of the search space in the long run. The forager selects a random food site and then based on its analysis of the population topology; it selects the *p*-fittest site (*p* being 20%) along with two random members. Its motion has two components one along the *p*-fittest member and the other along the direction vector pointing from one random site to another.

The associated vector representation for the movement trajectories are given as follows:-

$$1:\ \text{ABC}\text{---}\{sto\}\{1\}:\qquad v^C_{i,j} = \theta^C_{r_1,j} + \varphi_{i,j}\cdot\left(\theta^C_{r_2,j} - \theta^C_{r_3,j}\right) \tag{37}$$

$$2:\ \text{ABC}\text{---}\{src\}\{p-fit\}\{2\}\quad v^C_{i,j} = \theta^C_{i,j} + \varphi_{i,j}\cdot\left(\theta^C_{p_{fittest},j} - \theta^C_{i,j}\right) + \varphi_{i,j}\cdot\left(\theta^C_{r_1,j} - \theta^C_{r_2,j}\right) \tag{38}$$

$$3:\ \text{ABC}\text{---}\{sto\}\{p-fit\}\{2\}\quad v^C_{i,j} = \theta^C_{r_1,j} + \varphi_{i,j}\cdot\left(\theta^C_{p_{fittest},j} - \theta^C_{r_1,j}\right) + \varphi_{i,j}\cdot\left(\theta^C_{r_2,j} - \theta^C_{r_3,j}\right) \tag{39}$$

Here the perturbation factor $\varphi_{i,\,j}$ is sampled from the Levy distribution as

$$\varphi_{i,j} = Levy(\mu, \gamma(C)) \tag{40}$$

where $\mu$ and $\gamma$ are the location parameter and scale parameter respectively. $\gamma$ is updated deterministically as

$$\gamma(C) = \gamma(0) - (\Delta\gamma) \cdot \left(\frac{e^{m.C/MCN} - 1}{e^{m} - 1}\right) \tag{41}$$

where $\gamma(0) = 0.2$, $\mu = 0$, $\Delta\gamma = 0.15$ and $m = 7$. The deterministic variation of $\gamma$ with iteration coupled with the wider tail of Levy Distribution [23], results in generation of a wide range of values of $\varphi_{i,j}$ that will ensure greater perturbation in the position of the foragers, thereby reducing the chances of getting trapped in a local optimum.

The selection of a particular perturbation strategy from the above mentioned strategies is governed by a perturbation probability called $prob_{strategy}$, which is sampled from a Gaussian distribution as

$$prob_{strategy} = Gaussian(0.5, 0.3)$$

For selection of a particular strategy, two particular flag values are used 1/3 and 2/3, and the value of $prob_{strategy}$ is compared. If $prob_{strategy}$ lies in the range (0, 1/3] then ABC—{*src*}{p-*fit*}{2} is enacted. Similar range selection values for ABC—{*sto*}{1} and ABC—{*sto*}{p-*fit*}{2} are [1/3, 2/3) and [2/3, 1] respectively.

These employed bees continue to analyze the quality of the solutions in the vicinity of the food sources, which is based on the fitness values that in turn depends upon the nectar contents of the food sources. The fitness associated with the $i^{\text{th}}$ food source (for minimization problems) is calculated using Eq. (35).

The purpose of using the pool of perturbation strategies is to widen the search range of the forager and thus ensure a better exploration of solution space. The scale parameter deterministically controls the amount of exploration and exploitation permitting the former in earlier stages while promoting the latter during the final stages of convergence for better accuracy. This is followed by the greedy selection mechanism for choosing between the original food source and newly found one based on their fitness. It can be implemented as follows.

$$\vec{\theta}_i = \left\{ \begin{matrix} \vec{v}_i \text{ if } fit(\vec{v}_i) \geqslant fit(\vec{\theta}_i) \\ \vec{\theta}_i \text{ if } fit(\vec{v}_i) < fit(\vec{\theta}_i) \end{matrix} \right\} \tag{42}$$

## 6.2. Decentralized onlooker phase

In the classical ABC framework, the onlooker bees waiting in the hive select a food source for future exploitation by a probabilistic scheme called Roulette Wheel Selection and the selection process continues until all onlookers have been allotted food sources. Karaboga [19] stressed on the importance of *T*/*E* ratio (*T* stands for total reward associated with the food source and *E* is the Euclidean distance measured from the hive) for selection of food sources. But the onlooker bee phase in the original ABC algorithm does not implement this thereby resulting in trapping in the vicinity of local optima caused due to overcrowded onlookers. To circumvent this problem, our algorithm incorporates a decentralized tasking (*Dt*) scheme by considering both relative fitness and the distance of the food sources with respect to the present location of the onlooker.

The *Dt* scheme is biologically motivated from the fact that both the distance and reward are crucial to bees for selecting food source. Analytically it can be said that if the nectar rewarded is meager compared to the energy expended in traveling towards the nectar source, then the foragers are likely to discard the site. Favorable considerations are given to sites where sufficient reward is available. Since efficiency and decentralization is our main concern, we modify the motion of onlooker foragers as

$$v^{C}_{i,j} = \theta^{C}_{i,j} + \varphi_p \cdot \left(\theta^{C}_{i,j} - \theta^{C}_{k,j}\right) + \varphi_s \cdot \left(\theta^{C}_{best,j} - \theta^{C}_{worst,j}\right) \tag{43}$$

where $\vec{\theta}_{best}$ and $\vec{\theta}_{worst}$ are the best and worst food sources in the neighborhood adjudged as the ones having the maximum and minimum T/E values given below.

$$(T/E)_{i,j} = \frac{1}{\|\theta_i, \theta_j\|}\left(\frac{fit_j}{fit_i} - 1\right)_{k=1,2,\ldots,SN} \tag{44}$$

The scaling factors $\varphi_p$ and $\varphi_s$ being sampled randomly as given below by using formulae (33), (44) and (45)

$$\begin{aligned} \varphi_p &= \omega + (1-\omega).rand(0,1).\left(1 - \left|\frac{\vec{\theta}_i - \vec{\theta}_j}{\vec{\theta}_{\max} - \vec{\theta}_{\min}}\right|\right) \\ \varphi_s &= \omega + (1-\omega).rand(0,1).\left(1 - \left|\frac{\vec{\theta}_{best} - \vec{\theta}_{worst}}{\vec{\theta}_{\max} - \vec{\theta}_{\min}}\right|\right) \end{aligned} \tag{45}$$

with the value of $\omega$ being 0.35 and 0.25 for $\varphi_p$ and $\varphi_s$ respectively.

### 6.3. Discarding of redundant solutions

Following the classical ABC, a *trial* counter is used for discarding those solutions which are considered to be exhausted or do not show any signs of improvement. The bees which are associated with those food sources are randomly initialized becoming random scouts and start their movement in a randomized fashion in the search space. To determine whether a solution is to be abandoned or not, the trial counter is maintained for each solution and it is incremented each and every time the greedy selection mechanism fails to select a solution having better fitness value. When the trial value exceeds a certain pre-defined *limit*, the involved food source is initialized to a random location and the corresponding trial counter value is set to 0.

## 7. Experimental results

### 7.1. Filter design

The results of the two filter design problems i.e. Butterworth and state variable are tabulated in this section and the results of our algorithm CRbABC_*Dt* have been compared with the 6 peer algorithms as elucidated below. The standard parametric setting of these algorithms as mentioned in the corresponding works is taken.

(i) DE rand/1/bin [37].
(ii) jDE [6].
(iii) jADE [47].
(iv) PSO [22].
(v) Comprehensive Learning PSO (CLPSO) [24].
(vi) ABC [19,39].

Here all the results associated with the filter design problems are obtained after due consideration of the best possible combinations (series combination for resistors and parallel combination for capacitors) of all the values included within the respective series (E12, E24 and E96). Note that to make the comparison uniform, the same initial seed for the random number generator has been granted to all the contending algorithms for a particular run. The test runs of the competing algorithms have been simulated 50 times for assimilating the data. The specifications of the host platform used are stated below.

1. Language: MATLAB (version R2012a).
2. Processor: Intel (R) Core (TM) *i*5-2450 M.
3. Speed: 2.50 GHz.
4. Memory: 4.00 GB (usable 3.90).
5. Operating System: Microsoft Windows 7 Home Premium SP1.
6. System Type: 64-bit.

#### 7.1.1. Results of Butterworth filter design

The discrete component values of the resistors and the capacitors used in the Butterworth filter design are restricted to the E12 series with the target cost function result to be set at a value smaller than 0.018, as suggested in [39]. All the concerned algorithms have been run with their best possible parametric setup by considering the design constraints defined in Eqs. (23) and (24). The ideal values obtained are rounded off to lie within the E12 series. The results of the design process, including the total error as well as the actual values of the circuit components have been tabulated in Table 1. Here all the

**Table 1**
Values of circuit components associated with Butterworth filter design (Best entry is marked in boldface).

| | Conventional design | | DE (rand/1/bin) | jDE | JADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|---|---|
| | Ideal values | Nearest preferred | | | | | | | |
| $R_a$ (kΩ) | 1 | 1 | 42.52 | 1.1 | 2.8 | 4.58 | 3.96 | 4.7 | 9.2 |
| $R_b$ (kΩ) | 1 | 1 | 21 | 1.88 | 27 | 4.7 | 19.6 | 4.7 | 18.9 |
| $C_a$ (nF) | 38.27 | 39 | 1.2 | 26 | 2.5 | 8.2 | 3.2 | 8.2 | 2.7 |
| $C_b$ (nF) | 26.13 | 270 | 9.2 | 191 | 52 | 56 | 39 | 56 | 21 |
| $R_c$ (kΩ) | 1 | 1 | 2.22 | 6.1 | 2 | 1.1 | 3.5 | 1 | 11.50 |
| $R_d$ (kΩ) | 1 | 1 | 16.5 | 6.07 | 9.2 | 1 | 2.8 | 39 | 16.40 |
| $C_c$ (nF) | 92.39 | 100 | 9.7 | 15 | 16.5 | 87.6 | 29.2 | 4.7 | 6.6 |
| $C_d$ (nF) | 261.3 | 100 | 27 | 18 | 33 | 102.2 | 35 | 56 | 8 |
| $\Delta\omega$ | 0 | 0.02549 | 0.0282 | 0.0134 | 0.0097 | 0.0135 | 0.0167 | 0.0201 | 0.0092 |
| $\Delta Q$ | 0 | 0.05026 | 0.0057 | 0.0080 | 0.0246 | 0.0018 | 0.0040 | 0.0024 | 0.0030 |
| Total error | 0 | 0.03788 | 0.0169 | 0.0107 | 0.0171 | 0.0076 | 0.0103 | 0.0113 | **0.0061** |

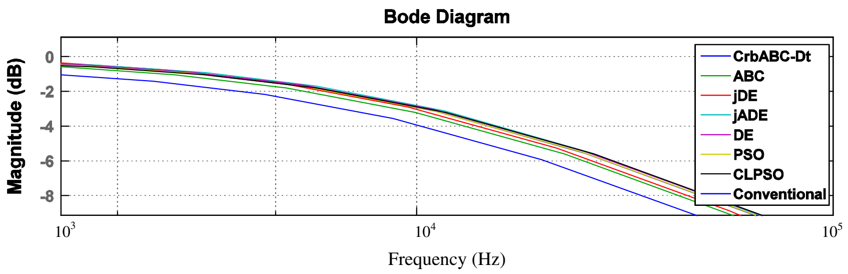


**Fig. 5.** Frequency response of fourth order low pass Butterworth filter (E12 series).

**Table 2**
Computation time associated with each algorithm in Butterworth filter design (Best entry is marked in boldface).

| Algorithms | DE (rand/1/bin) | jDE | jADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|
| Computation time (in s) | 43.1 | 33.6 | 10.5 | 24.3 | 29.2 | **2.1** | 2.8 |

competing algorithms are run until the desired functional value is obtained after approximating the values of various circuit components to their nearest E12 values. The associated computation times required to reach the desired CF value are listed as below for each algorithm:

The results show the superiority of CRbABC_*Dt* algorithms in obtaining the best minimal error associated with the design process i.e. 0.0061, in about 2.8 s.

The efficiency of the EC-based Butterworth filter synthesis is investigated by designing the Butterworth filter with the corresponding E12 series compatible values of resistors and capacitors, corresponding to each algorithm along with the LM 741 Opamp macro model in MultiSim 11.0 and obtaining the associated frequency responses. The frequency response of the associated algorithms is shown in Fig. 5, where the vertical axis is marked in decibels (dB) and the horizontal axis in terms of frequency (Hz).

### *7.1.2. Results of state-variable filter design*

The associated discrete component values of the resistors and capacitors considered in the SVF design are derived from the E24 and E96 series of values. The target cost function value as proposed by Vural et al. [39] was set at a value smaller than $1 \times 10^{-3}$. The design constraints are given in (28)–(30). The ideal values obtained as a result of application of EC techniques are rounded off to lie within the E24 and E96 series by determining their appropriate combinations (series combination for resistors and parallel for capacitors). (see Table 2).

The results of Table 3 clearly indicate that when E24 series resistor values are considered, our proposed algorithm CRbABC_*Dt* was able to outperform all the competing algorithms in obtaining the minimal error, associated with the design process i.e. $2.908 \times 10^{-4}$, in about 5.7 s. The computation times required to reach the desired tolerance value are given in Table 4. When the issue of time is taken into account, ABC is the fastest requiring only 3.4 s to obtain the target cost function value.

In Tables 3 and 5, the ideal values of the discrete components, obtained as a result of cost function, are first rounded to the nearby values compatible with the E24 and E96 series respectively by selecting a particular value from the available set or a nearby value is chosen after considering proper combinations of the members of the respective series. For example, the

**Table 3**
Values of circuit components for state variable filter design (E24 series) (Best entry is marked in boldface).

| | Conventional design | | DE (rand/1/bin) | jDE | JADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|---|---|
| | Ideal values | Nearest preferred | | | | | | | |
| $R_a$ (kΩ) | 4 | 4.12 | 9.57 | 196 | 141 | 52.5 | 10 | 62 | 81 |
| $R_b$ (kΩ) | 1.656 | 1.69 | 5.9 | 20 | 70.5 | 8.06 | 1.65 | 1 | 29.3 |
| $R_c$ (kΩ) | 4 | 4.12 | 6.875 | 43.33 | 25.8 | 12 | 30.11 | 91 | 5.03 |
| $R_d$ (kΩ) | 4 | 4.12 | 229.66 | 238.62 | 162.7 | 98.5 | 212 | 4.3 | 95.7 |
| $R_e$ (kΩ) | 4 | 4.12 | 30.12 | 81.8 | 51.35 | 7.48 | 1.039 | 27 | 8 |
| $R_f$ (kΩ) | 4 | 4.12 | 39 | 6.6 | 8.80 | 10.39 | 3.9 | 1.8 | 32.1 |
| $C_a$ (nF) | 25 | 25.5 | 50 | 5.1 | 6.7 | 75.5 | 470 | 2.7 | 130 |
| $C_b$ (nF) | 25 | 25.5 | 5.7 | 20 | 20.8 | 14 | 37 | 3.6 | 5.7 |
| $\Delta\omega$ | 0 | 0.049 | $1.1 \times 10^{-3}$ | $6.245 \times 10^{-4}$ | $5.28 \times 10^{-4}$ | $3.880 \times 10^{-4}$ | $4.082 \times 10^{-4}$ | $1.434 \times 10^{-4}$ | $1.8177 \times 10^{-4}$ |
| $\Delta Q$ | 0 | 0.003 | $2.639 \times 10^{-4}$ | $2.388 \times 10^{-4}$ | $4.035 \times 10^{-4}$ | $1.936 \times 10^{-4}$ | 3.239 × 10-4 | $6.167 \times 10^{-4}$ | $7.9752 \times 10^{-5}$ |
| Total error | 0 | 0.026 | $6.795 \times 10^{-4}$ | $3.2418 \times 10^{-4}$ | $4.6579 \times 10^{-4}$ | $2.908 \times 10^{-4}$ | $3.661 \times 10^{-4}$ | $3.801 \times 10^{-4}$ | $\mathbf{1.8177 \times 10^{-4}}$ |

**Table 4**
Computation time for algorithms in state variable filter design (E24 series) (Best entry is marked in boldface).

| Algorithms | DE (rand/1/bin) | jDE | jADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|
| Computation time (in s) | 108 | 43.8 | 23.6 | 25.0 | 37.0 | **3.4** | 5.7 |

**Table 5**
Values of circuit components for state variable filter design (E96 series) (Best entry is marked in boldface).

| | Conventional design | | DE (rand/1/bin) | jDE | JADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|---|---|
| | Ideal values | Nearest preferred | | | | | | | |
| $R_a$ (kΩ) | 4 | 4.12 | 16.2 | 165 | 107 | 48.4 | 10.2 | 59 | 21.475 |
| $R_b$ (kΩ) | 1.656 | 1.69 | 8.128 | 46.4 | 83.8 | 56.5 | 8.66 | 88.7 | 14.174 |
| $R_c$ (kΩ) | 4 | 4.12 | 5.69 | 13.7 | 357 | 4.32 | 14.7 | 54.9 | 14.515 |
| $R_d$ (kΩ) | 4 | 4.12 | 48.5 | 128.78 | 365 | 118 | 187 | 90.9 | 86.550 |
| $R_e$ (kΩ) | 4 | 4.12 | 47 | 2.61 | 10.5 | 9.67 | 1.13 | 10 | 41.474 |
| $R_f$ (kΩ) | 4 | 4.12 | 8.45 | 5.51 | 8.5 | 68.500 | 2.94 | 51.1 | 18.845 |
| $C_a$ (nF) | 25 | 25.5 | 9.53 | 219.75 | 7.64 | 95.59 | 464 | 7.5 | 7.15 |
| $C_b$ (nF) | 25 | 25.5 | 22.5 | 29.74 | 15 | 4.32 | 82.5 | 4.32 | 10.67 |
| $\Delta\omega$ | 0 | 0.049 | $4.60 \times 10^{-4}$ | 0.785 × 10-4 | $1.9362 \times 10^{-4}$ | $7.0779 \times 10^{-4}$ | $1.457 \times 10^{-4}$ | 0.295 × 10-4 | $0.0864 \times 10^{-4}$ |
| $\Delta Q$ | 0 | 0.003 | $5.625 \times 10^{-4}$ | $5.2801 \times 10^{-4}$ | $2.4124 \times 10^{-4}$ | $0.6545 \times 10^{-4}$ | $4.759 \times 10^{-4}$ | 0.047 × 10-4 | $0.0749 \times 10^{-4}$ |
| Total error | 0 | 0.026 | $5.113 \times 10^{-4}$ | $3.0194 \times 10^{-4}$ | $2.1743 \times 10^{-4}$ | $3.8662 \times 10^{-4}$ | 3.108 × 10-4 | 0.171 × 10-4 | $\mathbf{0.0807 \times 10^{-4}}$ |

**Table 6**
Computation time for algorithms in state variable filter design (E96 series) (Best entry is marked in boldface).

| Algorithms | DE (rand/1/bin) | jDE | jADE | CLPSO | PSO | ABC | CRbABC_*Dt* |
|---|---|---|---|---|---|---|---|
| Computation time (in s) | 74 | 48 | 27.8 | 28 | 93.6 | **4.6** | 5.5 |

resistance value of $R_d$ under the column CRbABC_*Dt* in Table 5, i.e. 86.55 kΩ is obtained as a series combination of the E96 resistors 84.5 kΩ and 2.15 kΩ. Similarly, under the same column, capacitor value $C_b$ is obtained by the parallel combination of 2.80 and 7.87 capacitances. The associated computation times required to reach the desired threshold value are listed in Table 6.

When E96 series values are considered, CRbABC_*Dt* achieves least error, associated with the design process i.e. $0.0807 \times 10^{-4}$, in about 5.5 s. Only ABC required comparatively less execution time.

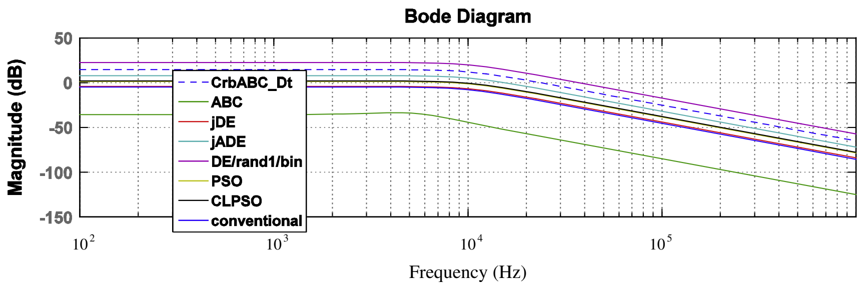


**Fig. 6.** Frequency response of a 2nd order state variable low pass filter (E24 series).

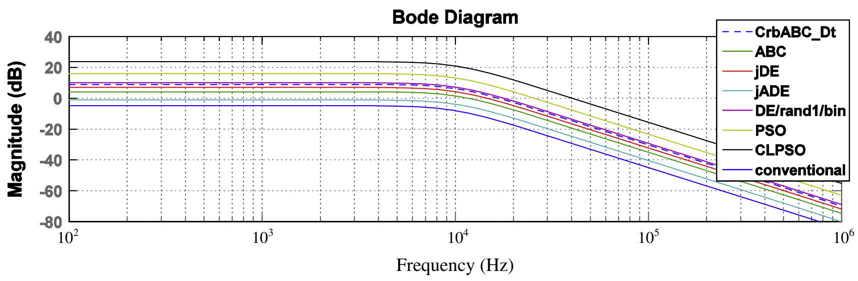


**Fig. 7.** Frequency response of a 2nd order SVF (low pass) (E96 series).

The results of the EA based state variable design are verified by designing the filter with the corresponding E24/E96 compatible values of resistors and capacitors, using a LM741 Opamp macromodel in MultiSim 11.0 and obtaining the associated frequency responses. The frequency response curves corresponding to E24 and E96 series are given by Figs. 6 and 7 respectively.

### 7.3. Results obtained on CEC 2013 benchmark

The performance of CRbABC_*Dt* has been compared with state-of-the-art on the most recent CEC 2013 benchmark proposed in the Special Session and Competition on Real Parameter Optimization [25]. The hybridized search behavior of CRbABC_*Dt* largely relies on the variation operators it uses. These islands themselves develop solutions centered on local optima, among which one may be closer to the global one. In promoting search, we aim at utilizing the useful information inherent in a solution to be used by another island model. Although the choice of algorithms is user-dependent, but the presence of a varied set of operators is generally beneficial while dealing with an extended problem set.

#### 7.3.1. Benchmark set

The theoretical benchmark set proposed in the CEC'13 Special Session and Competition on Real-Parameter Optimization is used to test the CRbABC_*Dt* method. It consists of 28 problems that can be grouped into 3 types.

- Unimodal Functions ($f_1$ to $f_5$).
- Basic Multimodal Functions ($f_6$ to $f_{20}$).
- Composition Functions ($f_{21}$ to $f_{28}$).

This set is an improvement over the previous CEC'05 benchmark set with some new inclusions and modifications. Interested readers are requested to see the technical report [25] for details.

#### 7.3.2. Algorithmic setup

The CRbABC_*Dt* is compared with algorithms: CLPSO [24], JADE [47], jDE [6] and modified ABC [2]. The parameters of the algorithms have been set as reported in literature. The same parameterization has been used regardless of the dimensions of the problems. Since CRbABC_*Dt* is a stochastic algorithm, several executions are required to properly analyze the scheme. Specifically, each run has been repeated 51 times.

**Table 7**
Mean and standard deviation of the best-of-the-run error values obtained for 10*D* functions (the best entries are marked in boldface).

| Func. | Algo | | | | |
|---|---|---|---|---|---|
| | *CRb*ABC_Dt | m-ABC | JADE | CLPSO | jDE |
| 1 | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) |
| 2 | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) |
| 3 | **2.283e−02 (1.301e−02)** | 6.315e+00 (2.103e+00) | 7.693e−01 (1.602e+00) | 6.302e+00 (2.098e+00) | 5.091e+03 (1.057e+03) |
| 4 | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 7.841e+03 (1.502e+03) |
| 5 | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) |
| 6 | **3.156e−02 (1.691e−01)** | 6.635e+00 (2.905e+00) | 2.748e−01 (7.455e−01) | 1.971e+00 (5.351e−01) | 3.831e+00 (1.294e+00) |
| 7 | **5.028e−03 (3.061e−02)** | 2.101e−02 (6.822e−02) | 3.061e−01 (1.602e+00) | 8.821e+00 (1.977e+00) | 1.111e+01 (2.392e+00) |
| 8 | 2.013e+01 (4.334e−02) | 2.014e+01 (4.667e−02) | 2.022e+01 (3.542e−01) | 2.038e+01 (1.251e−01) | 2.013e+01 (4.659e−02) |
| 9 | **8.503e−01 (8.355e−01)** | 9.168e+00 (9.410e−01) | 1.052e+00 (1.061e+00) | 6.898e+00 (2.037e+00) | 5.172e+00 (6.566e−01) |
| 10 | **0.000e+00 (0.000e+00)** | 3.473e−02 (8.706e−03) | 3.211e−03 (6.548e−03) | 6.064e−02 (1.382e−02) | 5.184e−02 (9.932e−03) |
| 11 | **0.000e+00 (0.000e+00)** | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) | 0.000e+00 (0.000e+00) |
| 12 | **1.526e+00 (6.884e−01)** | 1.202e+01 (1.924e+00) | 1.808e+00 (1.186e+00) | 5.982e+00 (1.589e+00) | 1.052e+01 (1.836e+00) |
| 13 | **1.454e+00 (7.302e−01)** | 1.204e+01 (2.786e+00) | 2.552e+00 (2.817e+00) | 1.268e+01 (4.018e+00) | 1.951e+01 (3.601e+00) |
| 14 | **3.545e−05 (6.585e−06)** | 3.721e+00 (2.649e+00) | 5.516e+00 (6.692e+00) | 3.216e+01 (9.318e+00) | 6.345e−02 (1.247e−02) |
| 15 | **4.553e+02 (1.026e+02)** | 1.320e+03 (1.248e+02) | 4.920e+02 (1.070e+02) | 1.253e+03 (2.905e+02) | 7.341e+02 (1.156e+02) |
| 16 | **7.876e−03 (3.494e−07)** | 1.472e+00 (1.634e−01) | 1.151e−02 (2.171e−02) | 5.263e−01 (1.251e−01) | 1.328e+00 (2.426e−01) |
| 17 | **8.644e+00 (2.857e+00)** | 1.013e+01 (6.444e−03) | 9.728e+00 (2.341e+00) | 1.121e+01 (3.119e+00) | 1.007e+01 (1.405e+00) |
| 18 | **9.905e+00 (3.457e+00)** | 3.631e+01 (3.149e+00) | 1.053e+01 (3.103e+00) | 1.578e+01 (5.031e+00) | 2.347e+01 (1.927e+00) |
| 19 | **3.595e−01 (5.926e−02)** | 7.756e−01 (8.762e−02) | 3.782e−01 (7.045e−02) | 7.294e−01 (1.544e−01) | 3.777e−01 (3.376e−02) |
| 20 | **1.858e+00 (2.855e−01)** | 3.062e+00 (2.729e−01) | 1.903e+00 (3.421e−01) | 3.285e+00 (5.777e−01) | 3.131e+00 (3.245e−01) |
| 21 | **3.334e+02 (8.143e+01)** | 4.002e+02 (2.891e−13) | 4.002e+02 (2.891e−13) | 4.002e+02 (1.750e−13) | 4.045e+02 (1.089e+02) |
| 22 | 3.312e+01 (2.846e+01) | 2.864e+02 (6.217e+01) | 4.312e+01 (3.626e+01) | 2.307e+02 (8.207e+01) | **1.395e+01 (4.191e+00)** |
| 23 | **4.005e+02 (1.606e+02)** | 1.864e+03 (2.481e+02) | 5.196e+02 (2.299e+02) | 1.271e+03 (4.562e+02) | 9.434e+02 (1.603e+02) |
| 24 | 1.922e+02 (2.302e+01) | 2.281e+02 (1.998e+00) | **1.913e+02 (2.410e+01)** | 2.120e+02 (3.096e+01) | 2.134e+02 (1.870e+01) |
| 25 | 1.852e+02 (3.172e+01) | 2.056e+02 (2.485e+00) | **1.816e+02 (3.123e+01)** | 2.229e+02 (3.246e+01) | 2.213e+02 (1.213e+01) |
| 26 | 1.089e+02 (1.248e+01) | 2.000e+02 (4.125e+01) | 1.158e+02 (2.520e+01) | **1.070e+02 (1.061e+00)** | 1.557e+02 (8.346e+00) |
| 27 | **3.112e+02 (1.053e+01)** | 5.925e+02 (2.618e+01) | 3.217e+02 (2.957e+01) | 5.482e+02 (7.785e+01) | 4.745e+02 (2.971e+01) |
| 28 | **2.236e+02 (8.471e+01)** | 3.000e+02 (0.000e+00) | 2.618e+02 (7.778e+01) | 3.000e+02 (9.093e+01) | 3.000e+02 (9.653e+01) |

**Table 8**
Mean and standard deviation of the best-of-the-run error values obtained for 30*D* functions (the best entries are marked in boldface).

| Func. | Algo | | | | |
|---|---|---|---|---|---|
| | *CRb*ABC_Dt | m-ABC | JADE | CLPSO | jDE |
| 1 | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) |
| 2 | 0.000e+000 (0.000e+000) | 8.578e+004 (1.837e+004) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 1.670e+003 (2.768e+002) |
| 3 | **3.525e+006 (5.313e+006)** | 2.122e+007 (3.874e+006) | 4.159e+006 (1.774e+007) | 1.828e+008 (4.051e+007) | 4.655e+007 (1.354e+007) |
| 4 | **1.173e+002 (2.489e+001)** | 1.877e+003 (3.479e+003) | 2.653e+003 (5.038e+003) | 1.383e+004 (5.359e+003) | 5.797e+004 (1.937e+004) |
| 5 | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) |
| 6 | **8.748e−001 (1.786e+000)** | 2.564e+001 (8.540e+000) | 3.488e+001 (6.045e+000) | 1.649e+001 (7.235e+000) | 7.713e+001 (1.319e+001) |
| 7 | **1.137e+001 (4.719e+000)** | 2.133e+001 (5.968e+000) | 1.366e+001 (7.313e+000) | 4.627e+001 (1.373e+001) | 1.138e+002 (2.181e+001) |
| 8 | 2.088e+001 (3.335e−002) | 2.103e+001 (4.789e−002) | 2.089e+001 (4.107e−002) | 2.098e+001 (5.279e−002) | 2.098e+001 (6.169e−002) |
| 9 | **1.574e+001 (1.766e+000)** | 4.152e+001 (1.293e+000) | 1.611e+001 (1.766e+000) | 2.661e+001 (4.746e+000) | 3.052e+001 (1.617e+000) |
| 10 | **3.133e−004 (8.469e−004)** | 1.626e−001 (4.494e−002) | 1.849e−003 (3.838e−003) | 1.232e−002 (4.406e−003) | 9.118e−002 (2.273e−002) |
| 11 | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) | 0.000e+000 (0.000e+000) |
| 12 | **1.181e+001 (2.526e+000)** | 7.428e+001 (1.512e+001) | 1.223e+001 (3.128e+000) | 2.885e+001 (5.608e+000) | 7.161e+001 (8.943e+000) |
| 13 | **2.060e+001 (9.082e+000)** | 1.277e+002 (1.941e+001) | 2.290e+001 (1.292e+001) | 9.000e+001 (2.320e+001) | 1.227e+002 (1.644e+001) |
| 14 | **6.246e−002 (1.846e−002)** | 4.646e+002 (7.181e+001) | 1.316e+002 (4.077e+001) | 3.766e+002 (1.027e+002) | 1.067e+002 (3.251e+001) |
| 15 | 4.096e+003 (1.011e+003) | 8.107e+003 (3.627e+002) | 4.097e+003 (8.994e+002) | 6.304e+003 (9.455e+002) | **4.062e+003 (2.709e+002)** |
| 16 | **1.195e−001 (2.778e−001)** | 3.059e+000 (3.386e−001) | 2.648e−001 (7.843e−001) | 2.464e+000 (8.495e−001) | 2.862e+000 (5.867e−001) |
| 17 | 3.233e+001 (5.275e−001) | 3.618e+001 (9.016e−001) | 3.242e+001 (7.401e−001) | 3.549e+001 (1.232e+000) | **3.043e+001 (1.351e−014)** |
| 18 | **4.092e+001 (3.101e+000)** | 1.757e+002 (1.125e+001) | 4.328e+001 (8.483e+000) | 8.293e+001 (1.228e+001) | 1.002e+002 (7.365e+000) |
| 19 | **1.498e+000 (2.596e−001)** | 4.742e+000 (4.115e−001) | 1.655e+000 (4.363e−001) | 2.840e+000 (5.132e−001) | 1.535e+000 (1.024e−001) |
| 20 | **1.037e+001 (8.122e−001)** | 1.286e+001 (4.217e−001) | 1.061e+001 (9.246e−001) | 1.219e+001 (1.039e+000) | 1.344e+001 (6.120e−001) |
| 21 | **1.935e+002 (3.383e+001)** | 4.435e+002 (7.772e+001) | 2.025e+002 (5.189e+001) | 2.000e+002 (2.945e+001) | 4.435e+002 (6.948e+001) |
| 22 | **3.038e+002 (9.879e+001)** | 3.168e+003 (5.622e+002) | 3.708e+002 (1.586e+002) | 9.201e+002 (3.034e+002) | 1.476e+002 (2.521e+001) |
| 23 | **4.347e+003 (6.123e+002)** | 4.616e+003 (3.229e+002) | 7.689e+003 (1.081e+003) | 8.264e+003 (1.401e+003) | 4.866e+003 (3.927e+002) |
| 24 | **2.233e+002 (4.977e+000)** | 3.159e+002 (3.459e+000) | 2.851e+002 (6.048e+000) | 2.601e+002 (8.897e+000) | 2.798e+002 (3.887e+000) |
| 25 | **2.865e+002 (3.490e+000)** | 3.045e+002 (3.112e+000) | 2.971e+002 (4.238e+000) | 2.997e+002 (6.729e+000) | 3.025e+002 (1.178e+001) |
| 26 | **2.005e+002 (2.306e+000)** | 4.069e+002 (7.580e+001) | 2.103e+002 (2.982e+001) | 3.335e+002 (4.474e+001) | 2.043e+002 (7.803e−001) |
| 27 | **5.700e+002 (6.514e+001)** | 1.416e+003 (2.499e+001) | 6.744e+002 (5.641e+001) | 9.507e+002 (1.640e+002) | 1.110e+003 (4.742e+001) |
| 28 | 2.838e+002 (4.629e+001) | 3.000e+002 (2.801e−013) | **2.876e+002 (7.034e+001)** | 3.000e+002 (5.052e+001) | 3.000e+002 (0.000e+000) |

### *7.3.3. Performance criteria*

Given the actual optimum of the function $f(X^*) = f_i(o) = f_i^*$ and the global optimum $f_i^*$ found by an algorithm, the best-of-the-run error value $(f_{best} - f_i^*)$ is recorded when *MaxFEs* (equal to $D \times 10^4$) has been expended. We report the mean and standard deviation of the recorded error values for 51 sample runs of 10*D* and 30*D* functions of each problem in Tables 7 and 8 respectively.

CRbABC_*Dt* performs very competitively in tracking the global optimum of the unimodal problems ($F_1$ to $F_5$). This behavior is natural due to the presence of strong global algorithms. The synergism of the search moves allows a harmonious blending of exploration and exploitation contributing to the versatility of CRbABC_*Dt*, as observed in Table 7. However in cases of problems with global optimum is far away from the local optima (as in $F_{15}$), the error value achieved by CRbABC_*Dt* is higher in comparison to other problems.

## *7.4. Discussions*

### *7.4.1. Algorithmic analysis*

In this paper the effectiveness of our proposed algorithm CRbABC_*Dt* in the field of active filter design has been investigated in comparison to well-known optimization algorithms through two specific cases of active filter design namely Butterworth and state variable synthesis, where specific attention was given to two factors that is associated accuracy with the design and the corresponding execution time. Theoretical benchmarks have also been used for a fair comparison.

In the design problems, the component values (i.e. resistors and capacitors) have been taken from the standard series i.e. E12 in case of Butterworth filter synthesis and E24 and E96 in case of SVF design. In case of Butterworth filter synthesis our proposed algorithm CRbABC_*Dt* was able to obtain the smallest design error in comparison with other algorithms but ABC was able to outperform others in terms of computation time. But the difference between the execution times of ABC and our proposed approach is very small. The state variable synthesis problem took into account both the E24 and E96 series values. Similarly CRbABC_*Dt* attained greater accuracy by providing the smallest possible design error. When the performance of other algorithms in case of state variable synthesis are taken into account with respect to the variation with the choice of components from various series, it was observed that jDE and PSO exhibited nearly the same performance even if the component values were chosen from different series. In both the design problems ABC required the least possible execution time with CRbABC_*Dt* emerging second best in all the possible cases.

In the light of the analog filter design and CEC'13 benchmark, the performance of CRbABC_*Dt* is commendable. The local-search ability of the ABC algorithm is sufficiently strong and is slightly similar to the standard DE algorithm (i.e., DE/rand/1), except that ABC modifies only a single function parameter. In case of multimodal fitness landscapes (like the transfer function of the filters), the tradeoff between exploration and exploitation is mandatory. Although the presence of Levy distribution may appear to make the search agents wandering in solution space, the T/E ratio in the onlooker bee phase helps to restore balance. The onlooker bee phase brings about an increase in the computation time due to the calculation of the affinity matrix but marked improvement is achieved as compared to its classical predecessor. Thus we can argue that the slight increase in the computation time is the price paid for the significant betterment in the design.

Combining distance-based and fitness-based information from the best ($\vec{\theta}_{best}$) and worst ($\vec{\theta}_{worst}$) member helps to initially focus on exploration (when the agents are scattered) and later on exploitation (agents gathered in fitness basins). The perturbation factors $\varphi_p$ and $\varphi_s$ play two different role. While the former check the spread of a randomly selected member with respect to the search space bounds, the latter focuses on the spread of the distance between the best and worst member with respect to the same bounds. Needless to say that the value of $\varphi_p$ and $\varphi_s$ being initially small, as gathered from Eq. (45), rises and never diminishes to negligibly small value though the differential vectors $\left(\theta^C_{i,j} - \theta^C_{k,j}\right)$ and $\left(\theta^C_{best,j} - \theta^C_{worst,j}\right)$ does during the end stages of the run. However their values lie in the range of (0, 1) unlike the perturbation factor generated by Levy distribution.

Importantly the problem of stagnation [43] is never present in ABC since it does not use a constant value of scale (perturbation) factor, sampled from a uniform distribution in the range [−1, 1]. Again the single component perturbation is another important feature of ABC. However the ABC confronts a different problem. The value of $\varphi$ varying from positive to negative and vice versa sometimes leads to a to-and-fro motion of the forager's position. Now consider that a forager (position) has found a local optimum. But due to the to-and-fro variation of $\varphi$ and the rectilinear movement of bees (refer to Eq. (34)), it is unable to improve its position since on either side of the local optima there is low-fitness region and greedy selection prevents the forager to improve upon its position. Thus ABC suffers from the tendency to get trapped in local optima. Additionally the basic perturbation logic, as per Eq. (34), fails to perform well for non-separable functions due to

**Table 9**
The *P*-values obtained for Kolmogorov–Smirnov test and Shapiro–Wilk's test (bold entries indicate that the normality is satisfied).

| Func. | Algo | | | |
|---|---|---|---|---|
| | Kolmogorov–Smirnov test | | Shapiro–Wilk's test | |
| | CR*b*ABC_*Dt* | m-ABC | CR*b*ABC_*Dt* | m-ABC |
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | 0.0000 | 0.0112 | 0.0000 | 0.0174 |
| 3 | 0.0410 | 0.0241 | **0.0621** | 0.0372 |
| 4 | **0.2700** | 0.0335 | 0.0315 | 0.02784 |
| 5 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 6 | **0.0801** | **0.1269** | **0.0964** | **0.0886** |
| 7 | **0.3612** | **0.3040** | **0.2527** | **0.3663** |
| 8 | **0.2871** | **0.1796** | **0.2413** | 0.0156 |
| 9 | **0.0710** | 0.0403 | 0.0112 | 0.0351 |
| 10 | 0.0234 | **0.0870** | 0.0314 | **0.1042** |
| 11 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 12 | **0.1932** | **0.1601** | **0.2713** | **0.1844** |
| 13 | **0.3286** | 0.0141 | **0.3578** | 0.0165 |
| 14 | **0.7550** | **0.0561** | **0.7765** | **0.4799** |
| 15 | 0.0372 | 0.0324 | 0.0443 | 0.0475 |
| 16 | **0.1749** | **0.0870** | 0.0172 | **0.2432** |
| 17 | **0.0820** | **0.1167** | **0.2962** | **0.1135** |
| 18 | **0.0554** | 0.0452 | 0.0378 | 0.03761 |
| 19 | **0.2271** | **0.0634** | **0.1821** | **0.0546** |
| 20 | **0.3392** | **0.3651** | **0.6427** | **0.2246** |
| 21 | **0.0922** | 0.0271 | 0.0413 | 0.0209 |
| 22 | 0.0152 | 0.0144 | 0.0174 | **0.0557** |
| 23 | 0.0071 | 0.0061 | 0.0062 | 0.0087 |
| 24 | 0.0174 | **0.0760** | 0.0261 | 0.0451 |
| 25 | **0.1523** | **0.1742** | **0.1646** | **0.4910** |
| 26 | **0.4180** | 0.0262 | **0.3802** | **0.2336** |
| 27 | 0.0032 | 0.0024 | 0.0021 | 0.0017 |
| 28 | 0.0011 | 0.0035 | 0.0014 | 0.0022 |

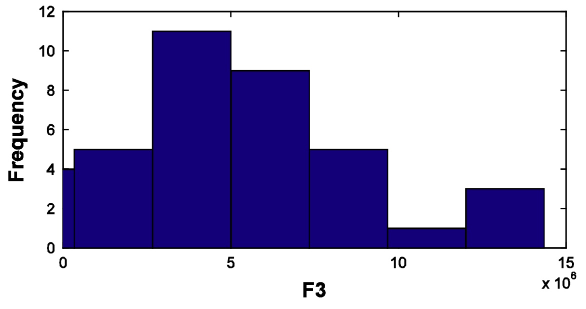
* In the table, the *P*-values which are less than 1e−08 are listed as 0.0000.

variable linkage. These issues are addressed by modifying the search logic that uses vector addition, see Eqs. (37)–(39), and restricting the scale factor to (0, 1) specially in the onlooker phase.

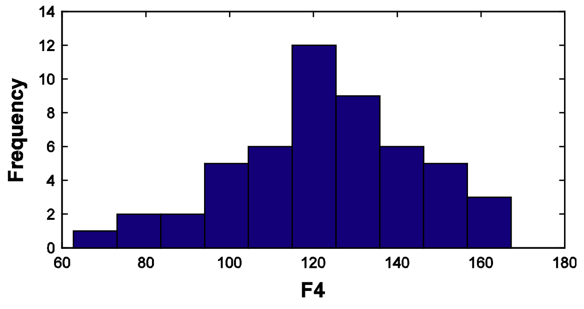
However, the ABC algorithm has a very successful decision mechanism that decides which areas within the search space require. Although basic ABC solely focused on fitness, the *Dt* scheme aids the foragers to judge a particular solution depending on the Euclidean distance (cost) associated with it. The probability generating mechanism is kept unaltered. The combined effort of these two approaches ameliorates the chance of finding fine solution maintaining tradeoff between exploration and exploitation. A similar PSO algorithm is based on the mathematical modeling of collective behavior of the living creatures that display complex social features. However, the stability problem in PSO restricts its success rate attaining high mean error values. In the PSO algorithm, while a particle is developing a new situation, both the cognitive component of the relative particle and the social component generated by the swarm are used and it enables the PSO algorithm to effectively improve the local solutions into global optimum solutions. But this has a negative effect in inducing oscillating effects on the particle when opposite attractive pulls are exerted. Unlike PSO, which can be significantly affected by the choice of initial parameter values, CRbABC_*Dt* is free from the value of weighting components used. The accuracy of DE variants used here suffers when a limited *FEs* budget is used and takes larger time to find solution due to the high degree of randomness and the lack of information sharing to some extent.

## 7.4.2. Statistical analysis

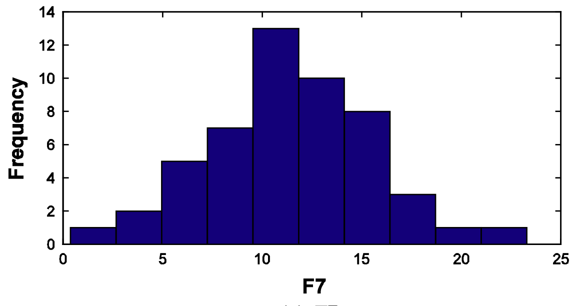
The mean and the standard deviation of the error values, which are tabulated in Tables 7 and 8 are considered for relevant statistical analysis. The mean of a dataset [44], which is primarily a single value, aims to describe the data set by locating a



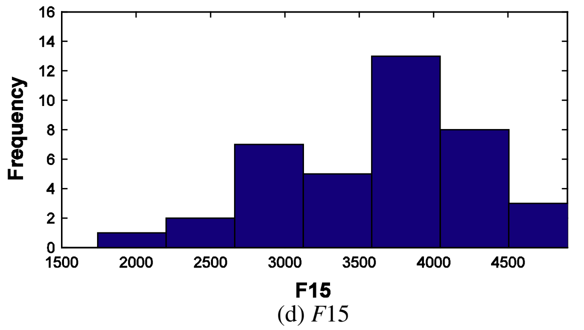
(a) *F*3



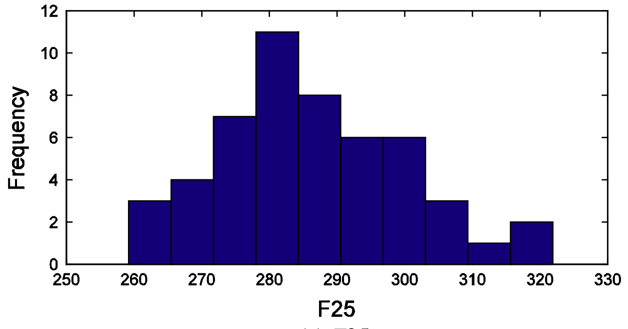
(b) *F*4



(c) *F*7



(d) *F*15



(e) *F*25

**Fig. 8.** Histogram plots of some CEC'13 benchmark functions.

suitable central position within the data and includes each member of the data set in the calculation. The inherent disadvantage of mean is the susceptibility to the presence of outliers in case of skewed frequency distribution i.e. the presence of certain members of the data set which are unusually small or large in comparison with others. For such skewed distribution another measure of central tendency i.e. median provides the best central location, since it is not affected strongly by the skewness in the data set.

The normal distribution [44], one of the most commonly occurring distribution in statistics is one particular case where all the measures of central tendencies i.e. mean, median and mode are equal. For testing whether a particular distribution is normal in nature, certain standardized tests like Kolmogorov–Smirnov test [14], Shapiro Wilk test [14], Liliefors test [44], etc. are used. We have used the Kolmogorov–Smirnov test and Shapiro Wilk test for determining whether an observation follows a normal distribution with a certain value of mean $\mu$ and standard deviation $\sigma$.

For both Kolmogorov–Smirnov and Shapiro Wilk tests, the significance value considered was $\alpha = 0.05$, so that a *P*-value greater than $\alpha$ ensures that the normality test is satisfied. Here for statistical analysis the algorithms CRb*ABC_Dt* and m-ABC are considered and for each function $F_1 - F_{28}$ in the CEC 2013 benchmark suite, the sample of results are obtained after running the algorithm 51 times. In the Table 9 the bold entries are used to denote that the normality is satisfied and the *P*-values for all the cases have been included.

In Fig. 8 the graphical representations in form of histograms are shown for certain functions of the CEC 2013 benchmark suite for the CR*b*ABC_*Dt* algorithm. Fig. 8(a) and (d), corresponding to the functions *F*3 and *F*15, represent the cases of non-normal distribution. The normal distribution nature is evident in the Fig. 8(b), (c) and (e), which is verified by the test results tabulated in Table 9.

## 8. Conclusions

The design of analog filter structures as a potential optimization problem have gained rapid prominence due to EAs' ability to perform as effective tools for selecting the best possible set of circuit components. This research article is one such attempt made by the authors to apply novel hybrid algorithmic variants in the design of highly efficient filters using two well-established techniques. A modified ABC variant is proposed here that tackles the observed short-comings of classical ABC and devises a novel algorithm that optimizes time and accuracy by furnishing marked improvement over prevailing algorithms.

As a future avenue of research, the authors would like to implement self-adaption for the development of certain algorithms which are capable of handling more complex filter structures, irrespective of the associated design limitations and maintaining suitable tradeoff between execution time and accuracy. Moreover, the use of Levy distribution adds a high degree of randomness which may be detrimental to certain class of multimodal functions like those used to benchmark niching algorithms [33]. The work may be extended in such direction by using the philosophy of Ockham's Razor in optimization. This can lead to future studies will to investigate the ABC functioning and redesign a neater and more efficient version.

## References

[1] B. Akay, D. Karaboga, Parameter tuning for the artificial bee colony algorithm, in: Computational Collective Intelligence: Semantic Web, Social Networks and Multiagent Systems, Wroclaw University of Technology; Swinburne University of Technology; Natl Taiwan University of Science and Technology, 2009, pp. 608–619 (Lecture Notes in Artificial Intelligence, 5796).

[2] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.

[3] L. Bao, J.C. Zeng, Comparison and analysis of the selection mechanism in the artificial bee colony algorithm, Int. Conf. Hybrid Intell. Syst. 1 (2009) 411–416.

[4] H.G. Beyer, E. Brucherseifer, W. Jakob, H. Pohleim, B. Sendhoff, T.B. To, Evolutionary Algorithms – Terms and Definitions, 2002. <http://ls11-www.cs.uni-dortmund.de/people/beyer/EA-glossary/,2002>.

[5] S. Biswas, S. Kundu, D. Bose, S. Das, P.N. Suganthan, B.K. Panigrahi, Migrating forager population in a multi-population artificial bee colony algorithm with modified perturbation schemes, in: Swarm Intelligence Symposium (SIS) Under Proceedings of IEEE Symposium Series on Computational Intelligence, Singapore, 16–19 April 2013, pp. 248–255.

[6] J. Brest, S. Greiner, B. Boškovi'c, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evolution. Comput. 10 (6) (2006) 646–657.

[7] A. Caponio, F. Neri, V. Tirronen, Super-fit Control Adaptation in Memetic Differential Evolution Frameworks, Soft Computing – A Fusion of Foundations, Methodologies and Applications, vol. 13 (8), Springer, 2009. pp. 811–831.

[8] S.-J. Chang, H.-S. Hou, Y.-K. Su, Automated passive filter synthesis using a novel tree representation and genetic programming, IEEE Trans. Evol. Comput. 10 (1) (2006) 93–100.

[9] X. Chen, Y-S. Ong, M-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–604.

[10] S. Das, A. Konar, A swarm intelligence approach to the synthesis of two-dimensional IIR filters, Eng. Appl. Artif. Intell. 20 (8) (2007) 1023–1162.

[11] S. Das, A. Konar, Design of two dimensional IIR filters with modern search heuristics: a comparative study, Int. J. Comput. Intell. Appl. 6 (3) (2006) 329–355.

[12] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.

[13] A. Das, R. Vemuri, An automated passive analog circuit synthesis framework using genetic algorithms, in: Proc. IEEE Computation Society Annual Symposium on VLSI, March 2007, pp. 145–152.

[14] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 Special Session on Real Parameter Optimization, J. Heurist. 15 (6) (2009) 617–644.

[15] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[16] D.H. Horrocks, M.C. Spittle, Component value selection for active filter using genetic algorithms, in: Proc. IEE/IEEE Workshop on Natural Algorithms in Signal Processing, Chelmsford, UK, vol. 1, 1993, pp. 13/1–13/6.

[17] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Dept, 2005.

[18] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Glob. Optim. 39 (3) (2007) 459–471.
[19] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[20] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. (2012), http://dx.doi.org/10.1007/s10462-012-9328-0.
[21] N. Karaboga, M. Cetinakya, A novel and efficient algorithm for adaptive filtering: artificial bee colony algorithm, Turk J. Elec. Eng. & Comp. Sci. 19 (1) (2011).
[22] J. Kennedy, R. Eberhart, Particle swarm optimization, Proc. of IEEE Int. Conf. on Neural Networks (1995) 1942–1948.
[23] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the levy probability distribution, IEEE Trans. Evol. Comput. 8 (1) (2004).
[24] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[25] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University and Tech Report, Nanyang Technological University, Singapore, January 2013.
[26] N.E. Mastorakis, I.F. Gonos, M.N.S. Swamy, Design of 2-D recursive filters using genetic algorithms, IEEE Trans. Circ. Syst. I 50 (5) (2003) 634–639.
[27] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Publication Report 790, Caltech Concurrent Computation Program, 1989.
[28] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evol. Comput. 2 (2012) 1–14.
[29] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. (2010), http://dx.doi.org/10.1007/s10462-009-9137-2.
[30] F. Neri, V. Tirronen, T. Kärkkäinen, T. Rossi, Fitness diversity based adaptation in multimeme algorithms: a comparative study, in: Proc. of the IEEE Congress on Evolutionary Computation, Special Session on Memetic Algorithms, Singapore, September 2007, pp. 2374–2381.
[31] Y-S. Ong, M.H. Lim, X. Chen, Memetic computation—past, present & future [research frontier], IEEE Comput. Intell. Mag. 5 (2) (2010) 24–31.
[32] L.D. Paarman, Design and Analysis of Analog Filters, Kluwer, Norwell, MA, 2007.
[33] B.Y. Qu, P.N. Suganthan, Novel multimodal problems and differential evolution with ensemble of restricted tournament selection, in: Proc. IEEE Congr. Evol. Comput, 2010, pp. 1–7.
[34] F.J. Rodriguez, C. García-Martínez, M. Lozano, Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test, IEEE Trans. Evol. Comput. 16 (6) (2012) 787–800.
[35] R. Schaumann, M.V. Valkenburg, Design of Analog Filters, Oxford Univ. Press, NewYork, 2001.
[36] A.F. Sheta, Analog filter design using differential evolution, Int. J. Bio-Inspired Comput. 2 (3–4) (2010) 233–241.
[37] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. (1997) 341–359.
[38] V. Tirronen, F. Neri, Differential evolution with fitness diversity self-adaptation, in: R. Chiong (Ed.), Nature-Inspired Algorithms for Optimization, Studies in Computational Intelligence, vol. 193, Springer, 2009, pp. 199–234.
[39] R.A. Vural, T. Yildrim, T. Kadiogluv, A. Basagran, Performance evaluation of evolutionary algorithms for optimal filter design, IEEE Trans. Evol. Comput. 16 (1) (2012) 135–147.
[40] R.A. Vural, T. Yildirim, Component value selection for analog active filter using particle swarm optimization, in: Proc. 2nd Intl. Conf. on Computer and Automation Engineering 1, 2010, pp. 25–28.
[41] R.A. Vural, T. Yildirim, State variable filter design using particle swarm optimization, in: Proc. 11th Int. Workshop Symbolic and Numerical Methods, Modelling and Applications to Circuit Design (SM2ACD), October 2010, pp. 1–4.
[42] W. Wang, Y. Lu, J. Fu, Y. Xiong, Particle swarm optimization and finite-element based approach for microwave filter design, IEEE Trans. Magnet. 41 (5) (2005) 1800–1803.
[43] M. 43, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Comput. 14 (11) (2010) 1187–1207.
[44] R.E. Walpole, R.H. Myers, S.L. Myers, K. Ye, Probability and Statistics for Engineers and Scientists, eighth ed. Pearson Education International, 2007.
[45] B. Xin, J. Chen, J. Zhang, H. Fang, Z-H. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, IEEE Trans. Syst. Man Cybernet. – Part C: Appl. Rev. 42 (5) (2012) 744–767.
[46] H. Xu, Y. Ding, Optimizing method for analog circuit design using adaptive immune genetic algorithm, in: Proc. Int. Conf. Frontier Comput. Sci. Technol., 2009, pp. 359–363.
[47] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 45–958.