


# Rational and self-adaptive evolutionary extreme learning machine for electricity price forecast

Chixin Xiao<sup>1,2,4</sup>  · Zhaoyang Dong<sup>3</sup> · Yan Xu<sup>3</sup> · Ke Meng<sup>3</sup> · Xun Zhou<sup>4</sup> · Xin Zhang<sup>3</sup>

Received: 12 September 2015 / Accepted: 3 June 2016 / Published online: 15 June 2016  
© Springer-Verlag Berlin Heidelberg 2016

**Abstract** Electricity price forecast is of great importance to electricity market participants. Moreover, various prediction approaches based on extreme learning machine (ELM) have been identified as effective on normal decision space. Especially, evolutionary extreme learning machine (E-ELM) may obtain better solution quality. However, in high dimensional space, E-ELM is time-consuming because it is difficult to converge into optimal region when just relied on stochastic searching approaches. In addition, due to the complex functional relationship is often complicated, the objective function of E-ELM seems hard to be mined directly for obtaining useful mathematical information to guide the optimum exploring. This paper proposes a new differential evolution (DE)-like algorithm to enhance E-ELM for more accurate and reliable prediction of electricity price. The approximation model for producing DE-like trail vector is the key mechanism, which may use simpler mathematical mapping to replace the original yet complicated functional relationship within a small region. Thus, the evolutionary

procedure frequently guided by rational searching directions may make the E-ELM more robust and faster than supported only by those stochastic methods. Several benchmarks are applied to test the performances of the proposed algorithm and the experimental results have shown that the new method can improve the performance of E-ELM.

**Keywords** Approximation model · Differential evolution · Extreme learning machine · Electricity price prediction

## 1 Introduction

In electricity industries, accurate price prediction [1] is very important for market participants to decrease risks along with their decision making. Historically, great research efforts have been devoted to developing accurate and reliable methods for electricity price prediction [2–5]. The state-of-the-art techniques include time-series methods such as Autoregressive Integrated Moving Average (ARIMA) [6,7] and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [8], and machine learning methods [9]. Among these techniques, machine learning models have shared the largest research attention mainly because of their strong nonlinear modeling capacity [10,11]. In respect of utilizing machine learning approaches for electricity price prediction as well as other applications in power system area, three main categories of the learning algorithms related to this paper are listed as follows,

- The first is the conventional machine learning tools, for instance, artificial neural networks (ANNs) [2,3], support vector machine (SVM) [10], etc. The best merit of such kind of approaches is that they can extract the nonlinear relationships out of the input and output dataset.

✉ Zhaoyang Dong  
zydong@ieee.org

Chixin Xiao  
chixinxiao@gmail.com

Xun Zhou  
zhouxun\_1978@hotmail.com

<sup>1</sup> The College of Information Engineering, Xiangtan University, Xiangtan 411105, China  
<sup>2</sup> The School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW 2522, Australia  
<sup>3</sup> The School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2006, Australia  
<sup>4</sup> The Centre for Intelligent Electricity Networks, University of Newcastle, Newcastle, NSW 2308, Australia

They therefore have been developed and adopted widely in power engineering domain during past decades. However, the approaches fallen in this category are mainly to use some dull gradient-descent information to guide the training for their forecasting models, and they are often deemed as lacking efficient strategies to escape premature convergence or many local minima.

- The other is a novel approach, ELM [12–15], which is proposed in recent years. ELM and its variants have one common distinguished capacity, fast training speed. At the beginning, they initialize hidden nodes randomly and then obtain the output weights via Moore–Penrose pseudo inverse [12], i.e. the output weights vector is treated as one solution of a group of linear-like equations. For many ordinary regression or classification problems being in low dimensionality, this kind of method is obviously enough to obtain better results than those traditional ones in both the training speed and training accuracy.
- Using evolutionary algorithms, E-ELM aims to evolve out an optimum result based on a population, usually in a constant size, consisting of ELM candidates for training. Significantly, ELM training is mainly depended on an unique step to find out the matrix solution instead of training step by step, say, the output weight vector is obtained on the basis of hidden layer output matrix and parameters assigned randomly. This is also the key factor why ELM is usually superior to those conventional approaches on training speed while still keeping good learning performance. However, calculating solution based on partly random assignments may lead to the final output unstable, sometimes with a better training effect, sometimes with an even worse one, e.g., as for high dimensional problems as in the works [1]. Besides that, ELM is sometimes difficult to find out a satisfied regression or classification results by once calculation, even though it can calculate output weights fast [16]. Overall, the difference of the solution quality provides the developing space for E-ELM.

Although the E-ELM can promote the final learning quality, it increases the time complexity as well. There are mainly due to the reasons in two aspects: First, until the final optimum to be found, the evolutionary algorithm has to calculate the ELM population rather than a single ELM individual iteratively by maximum generations. In addition, evolutionary algorithm itself also increases its time cost as the dimensionality of decision space expanding. According to [17], one random matrix of hidden weights is corresponding to a vector of output weights, besides that, better hidden weights and output weights can lead to a lower root mean square error, that is, a better regression or classification result for ELM. The hidden weights can be combined into a single row solution for the objective function of ELM, the optimum can be obtained

after evaluating the individual solution and comparing with other solutions among whole population. For example, let the data be 100 dimensions and the number of the hidden layers is 10, the dimensionality of the solution individual will then reach to 1000. Usually, the dimensionality of the power market data or the data for electricity load forecasting is over 100. Self-adaptive evolutionary extreme learning machine (SaE-ELM) [18] is a representative method of E-ELM, which can obtain output weights for a single hidden layer feed-forward network (SLFNs) with some promising features. However, in respect of training high dimensional data, SaE-ELM is also time-consuming in evolutionary iterations.

In order to promote the speed of E-ELM and robustness in optimum exploring in high dimensional space, this research is motivated to consider whether some rational analysis could be used to guide the optimum searching, especially in such environment as in high dimensional decision space, with a complicated objective function. As mentioned above, in conventional neural network, the gradient information provides some rapid exploring guides though often leading to local optima. So the gradient information perhaps can be properly used in E-ELM to provide some rational directions and then to accelerate the whole optimization procedure. Unfortunately, the complicated objective function of E-ELM is too difficult to mine the gradient information directly between decision variable vector and its fitness function. Moreover, the basic framework of ELM seems also to have pushed the gradient ideas out of date. Even so, the rest of this paper proposes a new approximation model, which not only can provide an approximate mapping dynamically to replace the old functional relationship of E-ELM within a comparative small region, but is compatible with differential evolution (DE) frame. Therefore, based on the new model, a hybrid DE-like algorithm is developed to ensure the new E-ELM not only can enter global optimal region faster than pure stochastic searching, but also can obtain high qualitative solutions, more reliably than those dull gradient methods. Overall, faster convergence and better quality of solution of E-ELM are two mandatory objectives should be considered in this paper.

The rest of paper is organized as follows. Section 2 outlines some relative backgrounds. The new approximation model is shown in Sect. 3. In Sect. 4, a new evolutionary algorithm for E-ELM learning high-dimensional data is proposed. The experimental results and discussions are placed in Sect. 5. Finally, a conclusion and future works are provided in Sect. 6.

## 2 Brief reviews

This section gives brief reviews of aboriginal extreme learning machine (ELM) as well as some necessary methods used in the rest of this paper for completeness.

### 2.1 Extreme learning machine (ELM)

ELM has been shown many good performances on the generalized single-hidden layer feed-forward networks (SLFNs) since it was proposed. It has some differences from the traditional neural networks on the hidden layer, e.g., random generation of its hidden nodes is the main feature of ELM. The basic working mechanism of ELM is briefly generalized as follows,

Given  $N$  training samples  $\{(x_i, t_i)\}_{i=1}^N$  which can be also described in matrix style  $\{(P, T_{tar})\}$ , where  $P$  is a  $D \times N$  real matrix of input data and  $T_{tar}$  represents  $N \times 1$  target vector.  $H$  is a  $L \times D$  real matrix consisting of the hidden layer parameters generated randomly.  $\beta$  is a  $L \times 1$  real vector of output weights. Their mathematical relationship can be expressed as Eq. (1)

$$f(H \cdot P + Bias)^T \cdot \beta = T_{tar} \tag{1}$$

where  $Bias$  is a  $L \times N$  real matrix and function  $f(\cdot)$  is a kind of activation functions [11], for instance, a log-sigmoid function,

$$\sigma(t) = \frac{1}{1 + e^{-c \cdot t}} \tag{2}$$

where  $c$  is a slop parameter.

Usually, Eq. (1) can be presented in brief as Eq. (3)

$$H \cdot \beta = T_{tar} \tag{3}$$

where  $H = f(H \cdot P + Bias)^T$  is a  $N \times L$  matrix. ELM uses Moore-Penrose pseudoinverse  $\hat{H}^\dagger$  and target vector  $T_{tar}$  to obtain a least-square solution of such linear system as Eq. (3). That is, a least-square solution of output weight vector  $\beta$  can be analytically determined as Eq. (4)

$$\hat{\beta} = \hat{H}^\dagger \cdot T_{tar} \tag{4}$$

where

$$\hat{H}^\dagger = \begin{cases} H^T \cdot (\frac{1}{c} + H \cdot H^T)^{-1}, & N < L \\ (\frac{1}{c} + H^T \cdot H)^{-1} \cdot H^T, & L < N \end{cases} \tag{5}$$

and  $c$  is a trade-off constant, which can be referred to [12–15] for more details.

Instead of following traditional gradient descend approach, ELM minimizes training accuracy or the cost function Eq. (6) via the result gotten by the Eq. (4).

$$RSME = \sqrt{mse(H \cdot \hat{\beta} - T_{tar})} \tag{6}$$

where  $mse(\cdot)$  is the function to measure network performance as the mean of absolute errors.

### 2.2 Basic differential evolution framework

Differential evolution (DE) [19] has been applied to many practical engineering problems since it was proposed in 1995. Furthermore, the variants of DEs with enhanced search performance have been introduced in [21–23]. Especially for multimodal optimization, researchers tend to combine DE with other evolutionary methods, sometimes called hybrid DEs, so as to promote whole performance of the algorithm.

In classical DE framework, the remarkably simple trial vector generation scheme is a main character distinguished from other EAs. It processes a scaled difference of vectors originating from a fixed-size population of decision vectors. Usually, such three evolutionary operators as mutation, crossover and selection are included respectively. During the  $g$ -th generation and in the basic DE mutation, a trial vector  $u_{i,g}$  is produced by a crossover operation between old individual vector  $x_{i,g}$  and a mutated vector  $v_{i,g} = x_{r0,g} + F_i \cdot (x_{r1,g} - x_{r2,g})$ , where  $F_i (F_i > 0)$  is a scaling factor,  $x_{r0,g}, x_{r1,g}, x_{r2,g}$  are three independent decision vectors selected randomly from the whole population  $P = \{x_{1,g}, x_{2,g}, \dots, x_{NP,g}\}$ . in decision space. For each vector  $x_{i,g} \in P$  in turn, there is a corresponding trial vector  $u_{i,g}$  being generated. Each old vector  $x_{i,g}$  in  $P$  will not be replaced unless its trial vector  $u_{i,g}$  yields a better objective function value than itself. Consequently  $x_{i,g}$  is also called a target vector in literature. One can refer to [24] for more different crossover operators and more variants of DE in detail.

### 2.3 SaE-ELM

Self-adaptive evolutionary extreme learning machine (SaE-ELM) [18] is upgraded from DE-LM [25] and E-ELM [17]. It chooses trial vector generation strategies and some relative control parameters adaptively. Their common place is to explore the network input weights and hidden node biases of ELM aiming to get optimum of the network output weights. When training data set  $X_{D \times N}$ ,  $L$  hidden layers and an activation function  $f(\cdot)$  are given, the individuals to be evolved during the  $g$ -th generation can be coded into as following vector [18],

$\theta_{k,g} = (h_{11}^g, \dots, h_{1D}^g, h_{21}^g, \dots, h_{2D}^g, \dots, h_{L1}^g, \dots, h_{LD}^g, b_1^g, \dots, b_L^g)$ , where  $1 \leq k \leq NP$ ,  $NP$  is the population size,  $b_i^g, 1 \leq i \leq L$ , represents the bias value for the  $i$ -th hidden layer in  $g$  generations.

Based on the coding format, the parameters like  $H, Bias$  are obtained as follows,

$$H = \begin{bmatrix} h_{11}^g, \dots, h_{1D}^g \\ h_{21}^g, \dots, h_{2D}^g \\ \vdots \\ h_{L1}^g, \dots, h_{LD}^g \end{bmatrix}, P = X_{D \times N}, Bias = \begin{bmatrix} b_1^g \\ b_2^g \\ \vdots \\ b_L^g \end{bmatrix} \times J_{1 \times N} \tag{7}$$

where  $J_{1 \times N}$  is a one row and  $N$  columns matrix of ones. Then the corresponding fitness function is formulated as Eq. (8),

$$RSME = \sqrt{mse \left( f(H \cdot P_{test} + Bias)^T \cdot \hat{\beta} - T_{test} \right)} \quad (8)$$

where  $P_{test}$  and  $T_{test}$  are testing data set and testing target vector respectively.

The main aim of such kind of algorithms is to explore an optimum of  $H$  from population consisted of  $\theta_{k,g}$  ( $1 \leq k \leq NP$ ) during  $g_{max}$  generations. The individuals which can survive from  $g$  generations to the next must satisfy Eq. (9).

$$\theta_{k,g+1} = \begin{cases} u_{k,g}, & RMSE_{\theta_{k,g}} - RMSE_{\theta_{k,g+1}} > \varepsilon \cdot RMSE_{\theta_{k,g}} \\ u_{k,g}, & |RMSE_{\theta_{k,g}} - RMSE_{\theta_{k,g+1}}| < \varepsilon \cdot RMSE_{\theta_{k,g}} \\ & \text{and } \|\beta_{u_{k,g+1}}\| < \|\beta_{u_{k,g}}\| \\ \theta_{k,g}, & \text{Otherwise} \end{cases} \quad (9)$$

### 3 Approximation model

Sometimes traditional optimization approaches are demonstrated very useful for evolutionary exploring as long as they can be manipulated properly. However, in many situations, traditional optimization methods are difficult to be used directly due to the complex functional relationship. These stimulate a motivation to employ simpler approximation models to replace the original fitness function. For example, as shown in Fig. 1, suppose the functional relationship between variable  $x$  and its objective function  $f(x)$  is complicated, yet in another space variable vector  $(t_1, t_2)$  can map into the same image  $f(x)$  by an easier mapping rule  $g(t_1, t_2)$ , hence, if a mapping relationship is found between the space in where  $x$  is located and the other space where variable vector  $(t_1, t_2)$  is included, then the original problem can be simplified, because the complex functional relationship has been replaced by a simpler mapping model. In the following section, a simple first-order approximation model is proposed in order to imitate the compound mapping between variable data and their objective function. Perhaps, such kind of functional relationship based on the approximation model is not very accurate to replace the original one over whole hyper-plane, but within a limited region it can satisfy those practical demands [26, 27].

#### 3.1 First-order approximation model

Without loss of generality, a decision space can be formulated as a hyper-plane by one point attached with two vectors. Let  $x \in \mathbb{R}^n$  is an arbitrary point in decision space  $\mathbb{R}^n$  or the point can be denoted as a decision vector  $(x_1, x_2, \dots, x_n)^T$ ,  $L$  is

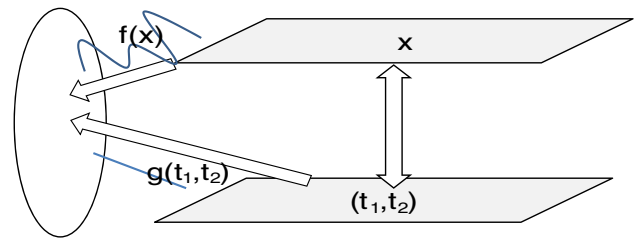


Fig. 1 Principle of the approximation model

the hyper-plane, suppose  $x^0 \neq x^1 \neq x^2$  are three distinct points selected randomly among  $\mathbb{R}^n$ , then any arbitrary point,  $x \in L$ , can be formulated as such style as Eq. (10)

$$x = x^0 + t_1 \cdot (x^1 - x^0) + t_2 \cdot (x^2 - x^0) \quad (10)$$

where  $t_1, t_2$  are two independent real variables.

According to Eq. (10), any  $x \in L$  is linear corresponding to the variable vector  $(t_1, t_2)$ , because the rest parameters are constants, i.e.,  $x \Leftrightarrow (t_1, t_2)$ , if and only if three arbitrary yet independent points,  $x^0 \neq x^1 \neq x^2$ , have been fixed. In other words, if  $x^0 \neq x^1 \neq x^2$  are located, any  $x \in L$  can be evaluated based on variable vector  $(t_1, t_2)$  and Eq. (10). Therefore, when decision vector  $x$  approaching its optimum,  $x^*$ , there must exist a corresponding variable vector  $(t_1^*, t_2^*) \Leftrightarrow x^*$ , i.e.,

$$x^* = x^0 + t_1^* \cdot (x^1 - x^0) + t_2^* \cdot (x^2 - x^0) \quad (11)$$

Likewise, for any pair of fitness function  $f(x)$  and its variable  $x$ , there exists another pair of image  $g(\cdot)$  and its variable vector  $(t_1, t_2)$  which has a common place, i.e.,  $g(t_1, t_2) = f(x)$ . However, the difference is the functional relationship of  $g(\cdot)$  is simpler than the one of  $f(\cdot)$ . The conversion relationship between  $g(\cdot)$  and  $f(\cdot)$  is defined as Eq. (12)

$$g(t_1, t_2) = f(x) = g^0 + t_1 \cdot (g^1 - g^0) + t_2 \cdot (g^2 - g^0) \quad (12)$$

where  $g^0, g^1, g^2$ , can be treated as constants, when  $x^0 \neq x^1 \neq x^2$  have been fixed as mentioned above. In order to obtain the constants,  $g^0, g^1, g^2$ , in a simple way, some special points are considered here. Assume  $(t_1, t_2)$  is substituted by vectors,  $(0,0), (1,0), (0,1)$  respectively, then  $g^0 = f(x_0), g^1 = f(x_1), g^2 = f(x_2)$  can be easily extracted out via Eqs. (12) and (10). Equation (12) hereby provides an approximation equation as well to replace the original fitness function, since  $g(t_1, t_2) = f(x)$ . So, no one would like to care about how the complicated functional relationship between the decision variable  $x \in \mathbb{R}^n$  and its original image  $f(x)$  is, as long as the new mapping between  $g(\cdot)$  and  $(t_1, t_2)$  is simpler and enough reliable.

### 3.2 Direction to optimum

As pointed out before, Eq. (12) also provides a linear functional relationship between variable vector  $(t_1, t_2)$  and its image  $g(t_1, t_2)$ . Through conventional optimization theories,  $g(t_1, t_2)$  at point  $(t_1, t_2)$  has a vector of first partial derivatives, or gradient vector  $\nabla g(t_1, t_2) = ((g^1 - g^0), (g^2 - g^0))$ . Hence, the local minimum optimum of  $(t_1^*, t_2^*)$  is most probably being placed in the opposite direction of  $\nabla g(t_1, t_2)$ .

$$(t_1^*, t_2^*) = (0, 0) - \alpha \cdot \nabla g(t_1, t_2) = -\alpha \cdot \nabla g(t_1, t_2) \quad (13)$$

where  $\alpha$  is a step parameter. Overall, any three distinct decision variables, say,  $x^0 \neq x^1 \neq x^2$ , can deduce out the local optimum  $x^*$  via Eqs. (11–13), which can be expressed as Eq. (14),

$$x^* = x^0 - \alpha \cdot \left[ (g^1 - g^0) \cdot (x^1 - x^0) + (g^2 - g^0) \cdot (x^2 - x^0) \right] \quad (14)$$

## 4 Proposed algorithm

In general, no method is flawless, neither is the new rational approximation model. Aiming to obtain a tradeoff between global exploration and local exploitation, another DE mutation strategies, ‘DE/current-to-best/1’ [24], is enrolled as well to construct a hybrid Rational and Self-adaptive mutation (RSM) strategy, whose pseudo-code is shown in Fig. 2.

### 4.1 Historical pool

From experimental results, fitness values evaluated by the new approximation model are very sensitive to the shape formed by three input individuals  $x_{r0,g}, x_{r1,g}$  and  $x_{r2,g}$ . The basic idea is to avoid excessive similarity between the candidates. Learn from JADE [21], RSM mutation applies a historical pool to temporarily reserve a part of individuals sifted out from the population. Each time, one of three distinct individual is picked out from the union of current population and the historical pool, hereby denoted by  $\hat{x}_{r2,g}$ , while the others  $x_{r0,g}, x_{r1,g}$  are still selected from the current population. The size of the historical pool is set to a quarter of the population and the initial state is empty. After being full, the pool permits the individual perished from current population to replace the worst one if the perished one is better.

### 4.2 Self-adaptive parameters

Motivated by [20–22], in the new algorithm many control parameters are extended into solution individuals for controlling self-adaptively (see Table 1). The parameters are evolved

```
function RSM-Trial()
input:  $x_{r0,g} \neq x_{r1,g} \neq \hat{x}_{r2,g}, x_{i,g}, x_{best,g}^P$  (one of the P best
individuals in current population,  $P=5$  in this paper)
output: two trial vectors  $u_{i,g}^1, u_{i,g}^2$ 
 $g^0 = f(x_{r0,g}); g^1 = f(x_{r1,g}); g^2 = f(\hat{x}_{r2,g});$ 
 $t_1 = -(g^1 - g^0); t_2 = -(g^2 - g^0);$ 
 $s = Step_{i,g} / \sqrt{t_1^2 + t_2^2};$ 
 $v_{i,g}^1 = x_{r0,g} + s \cdot [t_1 \cdot (x_{r1,g} - x_{r0,g}) + t_2 \cdot (\hat{x}_{r2,g} - x_{r0,g})];$ 
 $v_{i,g}^2 = x_{r0,g} + F_1 \cdot (x_{best,g}^P - x_{i,g}) + F_1 \cdot (x_{r0,g} - x_{r1,g});$ 
for j=1 to D
for k=1 to 2
if(j =  $k_{rand}$ ) or  $rand(0,1) < CR^k$ 
 $u_{j,i,g}^k = v_{j,i,g}^k$ 
else
 $u_{j,i,g}^k = x_{j,i,g}^k$ 
end if
end for
end for
end func
```

Fig. 2 Pseudo-code of producing hybrid trial vectors

Table 1 Encoding format of self-adapting individuals

$x_{1,g}$	$Step_{1,g}$	$CR_{1,g}^1$	$F_{1,g}$	$CR_{1,g}^2$
$x_{2,g}$	$Step_{2,g}$	$CR_{2,g}^1$	$F_{2,g}$	$CR_{2,g}^2$
...	...	...	...	...
$x_{NP,g}$	$Step_{NP,g}$	$CR_{NP,g}^1$	$F_{NP,g}$	$CR_{NP,g}^2$

simultaneously whilst the classical population of solutions is being processed in evolution procedure.

In general, the better control parameter value is corresponding to the optimal trial vector. Therefore the proposed algorithm utilizes the statistical results of recent successful parameters to guide the production of parameters for next generation.

Main parameters for self-adaptive control, such as  $Step_{i,g} \in [0, 2], CR_i^1, CR_i^2 \in [0, 0.9]$  and  $F_{i,g} \in [0.1, 1.0]$ , are initialized within their definition domain. The successful parameters survive to the next generation, while the unsuccessful ones are replaced by a normal distribution of the mean  $P_{m,g}$  and standard deviation  $sigma$  as shown in Eq. (15).

$$P_{i,g} = P_{m,g} + sigma \cdot randn_{i,g} \quad (15)$$

where  $P_{i,g}$  represents the variable of parameters for the  $i$ -th individual in  $g$  generation. The sigma of each parameter equals to  $\min(|P_{m,g} - P_{Ub,g}|, |P_{m,g} - P_{Lb,g}|)$ . The mean values are initialized as follows,  $Step_{m,1} = 1.1, F_{m,1} =$

0.6,  $CR_i^k = 0.6, (k = 1, 2)$ . Parameter  $Step_{i,g}$  controls the incremental degree of the mutation shown in Fig. 2.

$$v_{i,g}^1 = x_{r0,g} + s \cdot [t_1 \cdot (x_{r1,g} - x_{r0,g}) + t_2 \cdot (\hat{x}_{r2,g} - x_{r0,g})],$$

where  $s = Step_{i,g} / \sqrt{t_1^2 + t_2^2}$ . At the beginning of whole evolving procedure,  $Step_i \geq 1$  helps population converge to optimum fast, while  $Step_i < 1$  is good at effective exploitation, especially for solutions approaching to the optimum.

### 4.3 Hybrid strategy for trial vector

In the procedure of selection, as shown in Fig. 3, if two new trail vectors satisfies

– Case 1:  $f(x_{i,g}) < f(u_{i,g}^1) < f(u_{i,g}^2)$

Both two trail vectors are successful trail ones, i.e.,  $success(i,1) = 1, success(i,2) = 1$ , all their parameters can be kept to the next generation.

– Case 2:  $f(u_{i,g}^1) < f(x_{i,g}) < f(u_{i,g}^2)$

$u_{i,g}^1$  is named as a successful trail vector and  $success(i,1)$  is set to 1.

– Case 3:  $f(u_{i,g}^1) < f(u_{i,g}^2) < f(x_{i,g})$

This case means all the parameters need to be adjusted. At end of each generation, the mean of each parameter is adjusted by Eq. (16)

$$P_{i,g+1} = 0.85 \cdot P_{m,g} + 0.15 \cdot mean(P_{success,g}) \tag{16}$$

where  $mean(.)$  is a function of arithmetic mean.

### 4.4 RSM-DE algorithm

The main body of RSM-DE algorithm:

- Input:**  $NP$ : the size of the population;
- $Maxgen$ : the number of the maximum iteration;
- Fitness function;
- $D$ : The dimension of decision space.

**Output:** Optima of the fitness function.

#### Step 1 Initialization

Create a random initial population  $\{x_{i,0} | i = 1, \dots, NP\}$ . Initialize parameters within their definition regions.

**For**  $g = 1, \dots, Maxgen$ , **do**

#### Step 2 Evolution Items

**For**  $i = 1, \dots, NP$  **do**

```

function Selection()
input: two trial vectors  $u_{i,g}^1, u_{i,g}^2$ ;
output: two trial vectors  $x_{i,g+1}$  and successg;
success(i,1)g=success(i,2)g=0;
if(f( $u_{i,g}^1$ ) < f( $u_{i,g}^2$ ))
    if(f( $x_{i,g}$ ) < f( $u_{i,g}^1$ ))
         $x_{i,g+1} = x_{i,g}$ ;
    else if((f( $u_{i,g}^1$ ) ≤ f( $x_{i,g}$ )) && (f( $x_{i,g}$ ) ≤ f( $u_{i,g}^2$ )))
         $x_{i,g+1} = u_{i,g}^1$ ; success(i,1)g=1;
    else
         $x_{i,g+1} = u_{i,g}^1$ ; success(i,k)g=1; //k=1,2
    end if
end if
else if(f( $x_{i,g}$ ) < f( $u_{i,g}^2$ ))
     $x_{i,g+1} = x_{i,g}$ ;
else if((f( $u_{i,g}^2$ ) ≤ f( $x_{i,g}$ )) && (f( $x_{i,g}$ ) ≤ f( $u_{i,g}^1$ )))
     $x_{i,g+1} = u_{i,g}^2$ ; success(i,2)g=1;
else
     $x_{i,g+1} = u_{i,g}^2$ ; success(i,k)g=1; //k=1,2
end if
end if
end if
end func
    
```

**Fig. 3** Pseudo-code of selection operator

**Step 2.1 New Parameters Generating:** Unsuccessful parameters are refreshed based on Eq. (15).

**Step 2.2 Mating:** One of the  $P$  best individuals and other three independent individuals,  $x_{best,g}^P, x_{r0,g} \neq x_{r1,g} \neq \hat{x}_{r2,g}$ , are picked out.  $\hat{x}_{r2,g}$  is from the union of current population plus historical pool and  $x_{best,g}^P$  is one out of from current population,  $P = 5$  in this paper.

**Step 2.3 Call Function RSM-Trial():** To produce two trail vectors by two strategies respectively.

**Step 2.4 Call Function Selection():** To select successful trail vectors and parameters into the next generation.

**Step 2.5 Renew Historical Pool:** If the historical pool is not full then the eliminated individuals are pushed into the pool, otherwise the worst one in the pool is replaced when the eliminated one is better.

**Step 2.6 Summarize the Statistical Result of Successful Trail Vectors:** To evaluate the arithmetical mean value of each parameter by Eq. (16).

**Step 3 Stopping criteria** When stopping criterion is satisfied, the algorithm stops here and outputs corresponding results. Otherwise, goes to **Step 2**.

### 4.5 RSM-DE-ELM

Given a set of training data, a set of testing data, a candidate range for L hidden layers and an objective function  $g(\cdot)$ , RSM-DE-ELM algorithm is summarized as following Fig. 4. *RSM\_DE\_ELM* ( $\cdot$ ) represents a procedure to optimize ELM based on the RSM-DE algorithm. It returns the optimum of the net.  $[L_1, L_2]$  is the candidate range and *Train*, *Test* denote training set, testing set respectively.

## 5 Experimental results

In this section, being compared with other representative differential evolution approaches, the DE-like part of the

```

function Optimal_Layer()
input: [L1, L2], Train, Test
output: Optimum of Lbest, Fitnessmin
m1 = RSM_DE_ELM(Train, Test, L1);
m2 = RSM_DE_ELM(Train, Test, L2);
while(L1!=round((L1+L2)/2+0.1) or L1!=round((L1+L2)/2-0.1))
    m3 = RSM_DE_ELM(Train, Test, round((L2 + L1)/2 ));
if(m1>m2)
swap(m1, m2); swap(L1, L2);
endif
m2 = m3; L2 = round((L1 + L2)/2);
if(m1>m2)
swap(m1, m2); swap(L1, L2);
endif
endwhile
Lbest = L1; Fitnessmin = m1;
End func
    
```

Fig. 4 Pseudo-code of RSM-DE-ELM

proposed algorithm is first tested by several single objective benchmark, which are picked out from different test categories, in order to examine the capacity for exploring optimum. Furthermore, regarding performance of the new E-ELM method in machine learning area, the integrate algorithm is also utilized to the application for electricity price prediction. The experimental results are also compared against other state-of-the-art algorithms.

### 5.1 Performance on single objective benchmark

At the beginning of this section, six representative single objective benchmark listed in Table 2 are applied to testify the performance of RSM-DE algorithm. The indices of the benchmark are kept as same as [21] for convenient comparing. Function  $f_2, f_4$  are continuous unimodal functions,  $f_7$  is noisy quartic function,  $f_8, f_9, f_{12}$  are multimodal and the number of their local minima increase exponentially with the dimensionality of the problem. The main control parameters are set as: the population size is 100 when the decision space is 30 dimensions. JADE is one self-adaptive algorithm of state-of-the-art. In [21], JADE has shown better performances on many benchmark than SaDE [20], jDE [22] as well as PSO [28]. So this paper chooses JADE as one main reference to measure the new algorithm. Experimental results including RSM-DE, JADE [21], DE/rand/1 [19] are summarized in Table 3, simultaneously, Fig. 5 shows a group of convergence semi-log graphs for 30-dimensional problems with median values after 50 independent runs. Form these converging curves, it is easy to find the RSM-DE algorithm owning a very good performance in 30-dimensional space. Especially, the new algorithm is very successful on benchmark  $f_4$ , which shows clearly the approximation model works well both with the better rate of convergence and the more robust reliability. Even facing many local minima

Table 2 Six single objective benchmark

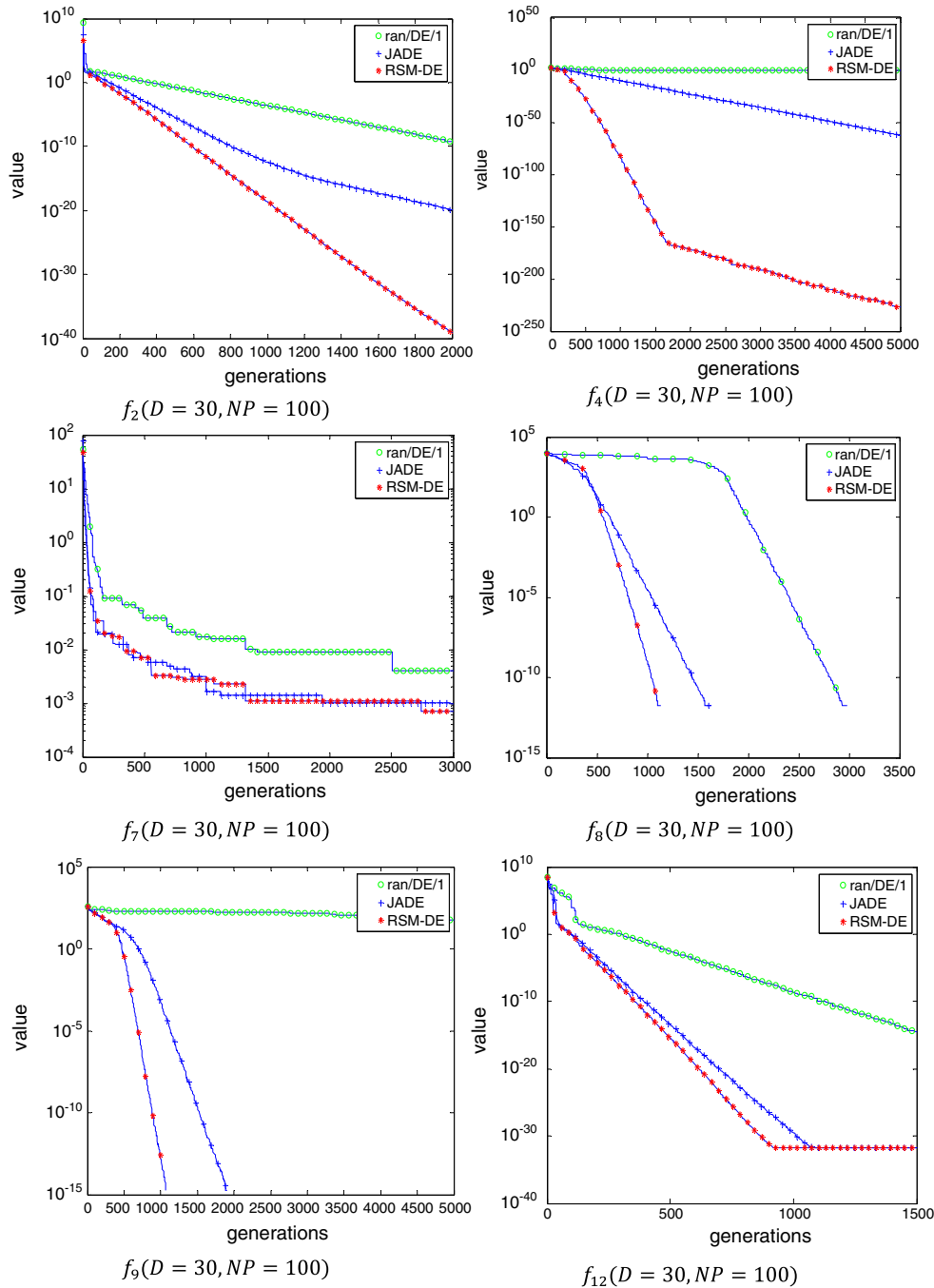
Objectives	Variable bounds
$f_2(x) = \sum_{i=1}^D  x_i  + \prod_{i=1}^D  x_i $	$[-10, 10]^D$
$f_4(x) = \max_i \{ x_i \}$	$[-100, 100]^D$
$f_7(x) = \sum_{i=1}^D i \cdot x_i^4 + rand[0, 1]$	$[-1.28, 1.28]^D$
$f_8(x) = \sum_{i=1}^D -x_i \cdot \sin \sqrt{ x_i } + D \cdot 418.98288727243369$	$[-500, 500]^D$
$f_9(x) = \sum_{i=1}^D x_i^2 - 10 \cdot \cos(2\pi x_i) + 10$	$[-5.12, 5.12]^D$
$f_{12}(x) = \frac{\pi}{D} \{10 \cdot \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \cdot [1 + 10 \cdot \sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^D u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1),$	$[-50, 50]^D$
$u(x_i, a, k, m) = \begin{cases} k \cdot (x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k \cdot (-x_i - a)^m & x_i < -a \end{cases}$	

**Table 3** Summary of QLD RRP from June 2006 to May 2007 (AUD/MWH)

	Winter	Spring	Summer	Autumn
Mean	26.5	22.01	43.99	61.18
Std.	75.91	13.93	229.76	56.69
Minimum	9.50	7.67	5.28	13.07
Median	19.00	18.77	22.68	53.34

Winter (Jun.–Aug.), Spring (Sep.–Nov.), Summer (Dec.–Feb.), Autumn (Mar.–May)

existed in the benchmark  $f_8, f_9, f_{12}$ , the RSM-DE still can accelerate the optimization procedure with promising results although its mutation operator is equipped with a greedy strategy. The results display if the diversity of population is being kept properly during the whole population evolving, the rational method will play a positive role rather than become a crucial problem.



**Fig. 5** Comparison of convergence curves between rand/DE/1, JADE, RSM\_DE on six representative benchmark



**Table 4** Six 30 dimensional benchmark with 50 independent runs

Func	Gen	RSM-DE Mean (std dev)	JADE Mean (std dev)	DE/rand/1 Mean (std dev)
$f_2$	2000	<b>9.7E-40</b>	1.8E-25	1.6E-14
		(2.5E-31)	(8.8E-25)	(1.1E-09)
$f_4$	5000	<b>3.7E-227</b>	4.3E-66	4.2E-01
		(0.0E+00)	(1.2E-65)	(1.1E+00)
$f_7$	3000	<b>5.2E-04</b>	6.4E-04	5.9E+03
		(5.1E-04)	(2.5E-04)	(1.1E+03)
$f_8$	1000	<b>1.2E-11</b>	3.3E-05	5.7E+01
		(6.8E-08)	(2.3E-05)	(7.6E+01)
	9000	0.0E+00	0.0E+00	5.7E+01
		(0.0E+00)	(0.0E+00)	(7.6E+01)
$f_9$	1000	<b>7.1E-12</b>	1.0E-04	1.8E+02
		(6.8E-10)	(6.0E-05)	(1.3E+01)
	5000	0.0E+00	0.0E+00	7.1E+01
		(0.0E+00)	(0.0E+00)	(2.1E+01)
$f_{12}$	500	<b>1.8E-17</b>	4.6E-17	1.2E-02
		(3.8E-16)	(1.9E-16)	(1.0E-02)
	1500	<b>1.5E-46</b>	1.6E-32	1.1E-14
		(2.1E-47)	(5.5E-48)	(1.0E-14)

### 5.2 RSM-DE-ELM for market price prediction

In this section, several sequential data series extracted from the Australian Energy Market Operator (AEMO) website [29] are used to test the performance of our new method.

For a convenient comparison, the first dataset in our case study is a whole year’s RRP’s from QLD market just as [1]. It includes total of 17520 observations and the period crosses over 01 June, 2006 to 31 May, 2007. Without loss generality, the dataset can be divided into four seasons in Australia and the main features are summarized as Table 4.

- The dataset format is constructed as follows,
  1. All observation points in every two weeks are defined as input attributes and their targets are those observation data from the following day, that is to say the data in the last day of each fifteen days is the target set and each day includes 48 even observation points.
  2. In each season, the data archive from the last seven successive days is used as the testing dataset. In order to compare efficiently, four other representative methods, BPNN, RBFNN, ELM and SaE-ELM are collected here to compare with the proposed approach via three different criteria, i.e., MAE, MAPE, and RMSE, respectively, which are listed in Eq. (17).

$$\begin{cases}
 MAE = \frac{1}{n} \sum_{i=1}^n |y_i - t_i| \\
 MAPE = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - t_i|}{y_i} \times 100 \% \\
 RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - t_i)^2}
 \end{cases} \tag{17}$$

- Parameter setting for RSM-DE-ELM:  
 The population size  $NP = 100$  and maximum generation is 60. Since the iteration number of RSM-DE is not high, so the mean values of the parameters are initialized as,  $Step_{m,1} = 1.1$ ,  $F_{m,1} = 0.7$ ,  $CR_{m,1}^k = 0.75$ , ( $k = 1, 2$ ). The candidate range  $[L_1, L_2]$  of hidden layers  $L$  is set to  $[10, 150]$ .

- Results analysis:

Table 5 shows the comparisons between the new approach and four existing methods [1, 18]. All these results are the mean values collected by multiple trails which include 50 independent forecasts of each season model. From Table 5 the new algorithm wins most of the lowest testing criteria in four season dataset among all these five approaches. For testing in Spring and Autumn, the performances have been improved dramatically. For example, the testing MAE of Spring using RSM-DE-ELM is 1.9443, while the other four methods, the testing MAEs of this season dataset are all greater than 2.2000.

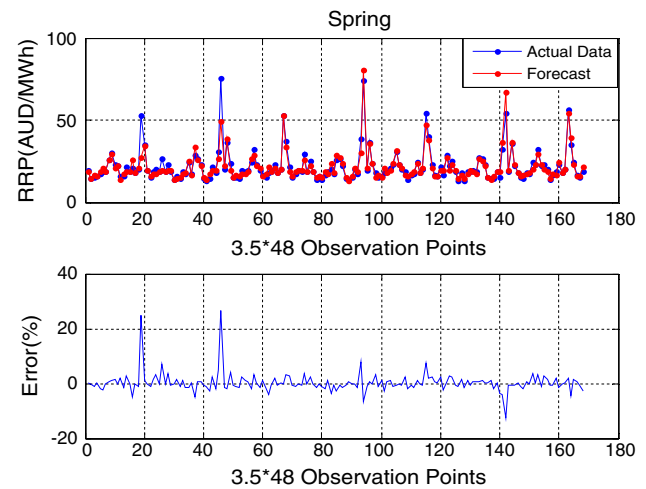
In Fig. 6., the prediction results is given out, which is run by RSM-DE-ELM on first half of  $7 \times 48$  observation points belonging to the testing dataset. The error curve shows the new algorithm can forecast with low and stable error rate in most points.

In terms of the training time, due to our approach falls into E-ELM category, the training procedure practically consists of several sub-trainings of basic ELM, thus it takes longer

**Table 5** Comparison of five methods on RRP Forecast

Season	Method	MAE	MAPE (%)	RMSE
Training				
Winter	BPNN	1.1566	5.1936	1.5225
	RBFNN	0.8345	3.8910	<b>1.1430</b>
	ELM	0.9458	4.3466	1.3544
	SaE-ELM	0.8843	3.4782	1.1768
	RSM-DE-ELM	<b>0.80145</b>	<b>2.9366</b>	1.1698
Spring	BPNN	1.3195	6.2802	1.7272
	RBFNN	<b>1.0048</b>	4.7648	<b>1.3311</b>
	ELM	1.2548	5.9897	1.6882
	SaE-ELM	1.2078	4.7655	1.5431
	RSM-DE-ELM	1.1896	<b>4.6761</b>	1.35257
Summer	BPNN	4.4040	15.3203	6.0542
	RBFNN	<b>3.1721</b>	11.4565	<b>4.3307</b>
	ELM	3.7803	12.3331	5.5882
	SaE-ELM	3.4862	9.01324	5.2344
	RSM-DE-ELM	3.2582	<b>7.4954</b>	4.4953
Autumn	BPNN	5.3081	9.6121	6.7923
	RBFNN	<b>3.9860</b>	7.6349	<b>5.1718</b>
	ELM	5.5116	10.0494	6.8658
	SaE-ELM	5.3983	8.7654	6.0065
	RSM-DE-ELM	5.1377	<b>7.43097</b>	5.7401
Testing				
Winter	BPNN	2.3611	9.9423	3.3470
	RBFNN	2.1046	8.5440	3.0537
	ELM	<b>2.0278</b>	8.3372	<b>2.9371</b>
	SaE-ELM	2.2789	7.6353	2.9922
	RSM-DE-ELM	2.0365	<b>6.0780</b>	2.9835
Spring	BPNN	2.2337	9.9291	3.1190
	RBFNN	2.6382	11.5712	3.6026
	ELM	2.3021	10.2642	3.1781
	SaE-ELM	2.2033	9.8776	<b>3.0332</b>
	RSM-DE-ELM	<b>1.9443</b>	<b>9.1967</b>	3.0549
Summer	BPNN	10.9983	24.4636	17.2313
	RBFNN	10.8783	22.7230	17.7526
	ELM	<b>10.1656</b>	21.8798	16.5881
	SaE-ELM	10.1966	21.472	16.3685
	RSM-DE-ELM	10.1797	<b>21.1376</b>	<b>16.2054</b>
Autumn	BPNN	7.8198	13.7900	10.7256
	RBFNN	7.9618	13.5447	11.4401
	ELM	7.3193	12.7363	10.3818
	SaE-ELM	7.3102	12.3147	10.3489
	RSM-DE-ELM	<b>7.0116</b>	<b>11.9539</b>	<b>10.2671</b>

time in training than one single basic ELM. However, the proposed approach is definitely faster than SaDE-ELM [18] because only two basic DE strategies are included rather than four in SaDE-ELM. Secondly, rational DE model provides

**Fig. 6** Average RRP of forecast by RSM-DE-ELM in spring

our method fast convergence in addition to promising experimental results, e.g., RSM-DE-ELM can get better results within 60 generations whilst SaDE-ELM need 100 more generations to reach the same magnitude. What's more, our approach is no longer running on the way mentioned in the previous literature [17, 18] in which the number of hidden layers is often gradually increased and the one with the best generalization performance is adopted in final. In our proposed approach, the binary search frame helps the algorithm not only find the optimum at last, but also keep in less time complexity.

## 6 Conclusion

In this paper, a self-adaptive DE-like frame embedded with a rational approximation operator is proposed intending to optimize E-ELM with faster speed and better solution. Based on the benchmark experimental results, it can be seen that the reliability of rational means become a less crucial problem in optimization, i.e., they are no more the patent of local optimum or premature convergence, on the contrary, their fast convergence becomes more attractive as long as a well-design scheme is provided. Furthermore, supported by the new rational approach, E-ELM has obtained better results than many state-of-art ones in the practical application of electricity price predication. Overall, the experimental results have illustrated that mathematical auxiliary guiding during evolving optima may create better performances for E-ELM than stochastic strategies did.

**Acknowledgements** This work is supported in part by the Australian Research Council (ARC) through a Linkage Project (Grant No. 120100302), in part by the University of Newcastle through a Faculty Strategic Pilot Grant, in part by the Research Foundation of Education Bureau of Hunan Province, China (Grant No. 14A136) and in part

by General Financial Grant 2015M572796 from the China Postdoctoral Science Foundation. The authors would like to thank Prof. Qingfu Zhang (UK & Hongkong) for his fruitful discussions and patient tutoring when I was in UK. Thank Dr. Jingqiao Zhang for providing the source code of JADE. Meanwhile, the authors wish to express gratitude to anonymous reviewers for their constructive comments.

## References

- Chen X, Dong ZY, Meng K, Xu Y, Wong KP, Ngan HW (2012) Electricity price forecasting with extreme learning machine and bootstrapping. *IEEE Trans Power Syst* 27(4):2055–2062
- Xu Y, Dong ZY, Xu Z, Meng K, Wong KP (2012) An intelligent dynamic security assessment framework for power systems with wind power. *IEEE Trans Ind Inf* 8(4):995–1003
- Meng K, Dong ZY, Wong KP (2009) Self-adaptive RBF neural network for short-term electricity price forecasting. *IET Gen Trans Dist* 3(4):325–335
- Wan C, Xu Z, Pinson P, Dong ZY, Wong KP (2014) Probabilistic forecasting of wind power generation using extreme learning machine. *IEEE Trans Power Syst* 29(3):1033–1044
- Amjady N, Keynia F (2009) Day-ahead price forecasting of electricity markets by mutual information techniques and cascaded neuroevolutionary algorithm. *IEEE Trans Power Syst* 24(1):306–318
- Contreras J, Espinola R, Nogales FJ, Conejo AJ (2003) ARIMA models to predict next-day electricity prices. *IEEE Trans Power Syst* 18(3):1014–1020
- Conejo AJ, Plazas MA, Espinola R, Molina AB (2005) Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. *IEEE Trans Power Syst* 20(2):1035–1042
- Garcia RC, Contreras J, Akkeren MV, Garcia JBC (2005) A GARCH forecasting model to predict day-ahead electricity prices. *IEEE Trans Power Syst* 20(2):867–874
- Li G, Liu CC, Mattson C, Lawarree J (2007) Day-ahead electricity price forecasting in a grid environment. *IEEE Trans Power Syst* 22(1):266–274
- Bishop CM et al (2006) *Pattern recognition and machine learning*, vol 1. Springer, New York
- Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. *Mach Learn* 3(2):95–99
- Li M-B, Huang G-B, Saratchandran P, Sundararajan N (2005) Fully complex extreme learning machine. *Neurocomputing* 68:306–314
- Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans Neural Netw* 17(4):879–892
- Huang GB, Chen L (2007) Convex incremental extreme learning machine. *Neurocomputing* 70(16–18):3056–3062
- Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. *Neurocomputing* 71(16–18):3460–3468
- Cao J, Lin Z, Huang G-B, Liu N (2012) Voting based extreme learning machine. *Inf Sci* 185(1):66–77
- Zhu Q-Y, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. *Pattern Recognit* 38(10):1759–1763
- Cao J, Lin Z, Huang G-B (2012) Self-adaptive evolutionary extreme learning machine. *Neural Process Lett* 36:285–305
- Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optim* 11(4):341–359
- Abbass HA (2002) The self-adaptive pareto differential evolution algorithm. In: *Evolutionary computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, pp 831–836
- Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. *Evol Comput IEEE Trans* 13(5):945–958
- Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *Evol Comput IEEE Trans* 10(6):646–657
- Brest, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans Evol Comput* 10(6):646–657
- Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. *Evol Comput IEEE Trans* 15(1):4–31
- Subudhi B, Jena D (2008) Differential evolution and Levenberg Marquardt trained neural network scheme for nonlinear system identification. *Neural Process Lett* 27(3):285–296
- Montgomery DC (2006) *Design and analysis of experiments*. Wiley, Hoboken, NJ
- Xiao C, Xue Z, and Yin J (2014) Rational models to improve performance of differential evolution for MOEA/D. In: *Natural computation (ICNC), 2014 10th International Conference on*, pp 335–342
- Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf Process Lett* 85(6):317–325
- Australian Energy Market Operator (AEMO), [www.aemo.com.au](http://www.aemo.com.au)