Orthogonal Learning Particle Swarm Optimization for Power Electronic Circuit Optimization with Free Search Range

Zhi-hui Zhan and Jun Zhang(Corresponding Author)
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Digital Life, Ministry of Education
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

***Abstract*—Power electronic circuit (PEC) always consists of a number of components such as resistors, capacitors, and inductors which have to be optimized in order to obtain good circuit performance. In current studies, the search ranges of these components are always pre-defined carefully by expert designers, making it difficult for practical applications. In this paper, the search space is freely set to the commonly used ranges and an efficient orthogonal learning particle swarm optimization (OLPSO) is applied to optimally design the PEC with such search space. OLPSO uses an orthogonal learning (OL) strategy for PSO to discover useful information that lies in the personal historical best experience and the neighborhood's best experience via orthogonal experimental design. Therefore, OLPSO can construct a more promising and efficient exemplar to guide particle to fly better towards the global optimal region. OLPSO is implemented to optimize the design of a buck regulator in PEC. The optimized results are compared with those obtained by using a genetic algorithm (GA) approach and those obtained by using PSO with traditional learning strategy. Results show that the OLPSO algorithm is more promising in the design and optimization of the PEC with large search space. Moreover, the simulations results demonstrate the advantages of OLPSO by showing that the circuit optimized by OLPSO exhibits better startup and large-signal disturbance performance when compared with the one optimized by GA.**

Keywords- Control systems; optimization; particle swarm optimization (PSO); orthogonal experimental design (OED)

I. INTRODUCTION

Power electronics has developed quickly since the advent of power semiconductor devices in the 1950s and has become a significant technology for variants of applications in the industrial, commercial, residential, aerospace, military, and utility areas [1]. The model, design, and analysis of power electronic circuit (PEC) are the fundamental and significant research areas in the power electronics [2]. PEC always consists of a number of components such as resistors, capacitors, and inductors which have to be optimized design in order to obtain good circuit performance [3]. Suitable components design and control parameters tuning of the PEC often challenge the engineers because they may require systematic procedure. Traditional approaches include the state-space average method [4], current injected equivalent circuit method [5], sampled-data modeling method [6], and state-plane analysis method [7], etc. However, these approaches are usually only applicable for specific circuits and require comprehensive knowledge on the circuit operation. Moreover, as these approaches are based on small-signal models, circuit designers would sometimes find it difficult to predict precisely the circuit responses under large-signal conditions.

With the rapid development of the power electronics technology and the growing complexity of PEC, automatic design and optimization of PEC have become great need. Since the 1970s, variants of optimization approaches such as heuristic method [8], knowledge based method [9], gradient descent or hill-climbing method [10], and simulated annealing method [11], have been proposed for analog circuit design automation. However, these approaches are very sensitive to the initial solution. Moreover, they might be inefficient enough to search globally and are subjective to be trapped into local optima when the problems are complex [16]. Therefore, the obtained values for the circuit components may be sub-optimal, leading to low satisfaction when used in practical applications.

Recently, evolutionary computation algorithms have fast developed and have been used as optimization techniques in many real-word applications, such as transportation scheduling [12][13] and network design optimization [14][15]. These optimization algorithms have played an important role in many science and engineering problems. Specifically, genetic algorithm (GA) [16] and ant colony optimization (ACO) [17] have been successfully applied to optimize the component values of PEC. However, there still exist disadvantages in the GA and ACO approaches that they have to consume a lot of computation before obtaining the good component values because of the slow convergence speed [18][19].

Moreover, previous studies using GA and ACO approaches always optimize the circuit components within careful pre-defined search ranges which are determined by expert designers [16]-[19]. For example, the search range for some resistors is from 470Ω to 47kΩ and the search range for some capacitors is from 0.33μF to 33μF [16]. However, such search ranges are difficult to be defined for different components in different PECs when used in practical applications. Therefore it

This work was supported in part by the National Natural Science Foundation of China (NSFC) No.61070004, by NSFC Joint Fund with Guangdong under Key Project U0835002

is of practical value to develop an effective and efficient approach that can optimize the components with free component configurations that are set to commonly used ranges. Nevertheless, the complex search space of PEC will challenge the efficiency of traditional approaches and requires optimization approach with strong global search ability. In this paper, an effective and efficient particle swarm optimization (PSO) algorithm, named orthogonal learning PSO (OLPSO) is adopted to optimize PEC because of its faster optimization speed and stronger global search ability [20][21]. PSO is a variant of evolutionary computation paradigm, featured with its simple concept and fast convergence speed, yet a very promising global optimization technique [22]. Even though PSO has been successfully applied to variants of practical problems, the exploration of PSO to PEC design and optimization is progressed at a slow pace, and only one algorithm based on the traditional PSO was proposed to design PEC in the literature [23]. However, traditional PSO is not efficient enough to solve complex problems because it is easy to be trapped into local optima [20][21]. Therefore, it is necessary to develop a powerful approach for the design and optimization of PEC.

The easiness of being trapped into local optima may be caused by the inefficient learning strategy of the traditional PSO in using the particle's historically best experience and the neighborhood's best experience [20][21]. In traditional PSO, the information of a particle's best experience and its neighborhood's best experience are used in a simple way, where the flying is determined by a simple summation of the two experiences. However, this is not necessarily an efficient way to make best use of the search information in these two experiences because it may cause an 'oscillation' phenomenon [24] and a 'two steps forward, on step back' phenomenon [25]. Hence, how to discover more useful information lying in the two exemplars and thus how to combine the information to construct an efficient guidance exemplar are significant and challenging research issues in PSO.

In order to obtain these goals, Zhan *et al*. [20][21] designed an orthogonal learning (OL) strategy for the particles to enhance their learning ability, thus the orthogonal learning PSO (OLPSO) is developed to enhance the PSO's performance. OLPSO uses an orthogonal experimental design (OED) [26][27] method to discover the best combination of a particle's best position and its neighborhood's best position in order to construct a promising learning exemplar. The orthogonal experimental factors are the dimensions of the problem and the levels of each dimension (factor) are the two choices of the particle's best position and its neighborhood's best position on the corresponding dimension. As OED offers an ability to discover the best combination levels for different factors with reasonably small number of experimental samples [26][27], the OL strategy is effective to discover the best combination of a particle's best position and its neighborhood's best position. This way, the best combination of the two exemplars can be constructed to guide the particle to fly more steadily, rather than oscillatory, because only one constructed exemplar is used for the guidance. It is also expected to be more promising towards the global optimum because the constructed exemplar makes the best use of the search information in both a particle's best position and its neighborhood best position.

Advantages of the OL strategy and the OLPSO algorithm have been demonstrated by comparing with PSO using traditional learning strategy on the test of mathematical benchmark functions [20][21]. As the OLPSO algorithm generally performs better on complex functions, this paper goes further to test its performance on solving the PEC problem. Simulation results on a buck regulator design are compared with those obtained by GA [16] and traditional PSO [23] approaches.

The rest of this paper is organized as follows. In Section II, the description of PEC and the implementation of OLPSO are presented. Then Section III proposes to use OLPSO to solve the PEC problem, including the particle representation, the fitness function, and the whole algorithm process. Section IV uses a buck regulator design example to verify the performance of OLPSO in optimizing the PEC. Simulation results are compared with those obtained by the GA and traditional PSO algorithms to verify effectiveness and efficiency of OLPSO. Finally, conclusions are summarized in Section V.

II. PEC AND OLPSO

A. Power Electronic Circuit

PEC is a circuit that contains a number of components such as resistors, capacitors, and inductors. Fig. 1 shows the basic block diagram of a PEC. In this PEC, the circuit can be decoupled into two parts where the first part named the power conversion stage (PCS) and the second part is the feedback network (FN) [16].

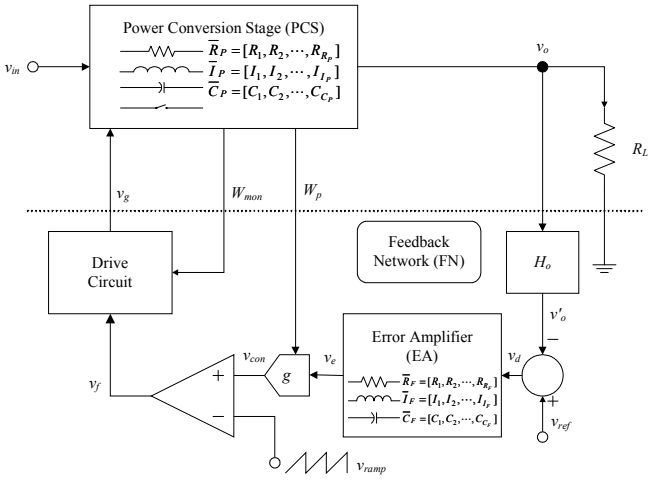

Figure 1. A block diagram of PEC.

The function of PCS is to transfers the power from the input source v_{in} to the output load R_L. It consists of R_P resistors, I_P inductors, and C_P capacitors. On the other hand, FN is the control part that consists of R_F resistors, I_F inductors, and C_F capacitors. There is a signal conditioner H in the FN circuit to convert the PCS output voltage v_o into a suitable form v'_o which is used to compared with the reference voltage v_{ref}. Their difference v_d is then sent to an error amplifier in order to obtain a output v_e. This output is combined with the feedback signals

W_p from the PCS part to give an output control voltage v_{con}. Then v_{con} is modulated by a pulse-width modulator to give a feedback voltage v_g to the PCS part.

In order to optimize the component values of PEC, the components are coded as the variables and are optimized through the optimization process. Although the components of PCS and FN can be optimized together in a single process, it is computationally intensive since the number of variables is considerably large. Moreover, as the interactions between the two parts in the optimization are relatively low during the training process, the components in the two parts can be optimized separately [16]. Therefore, this paper adopts the technique to consider the components of PCS and FN separately. This decoupled technique is not only effective to reduce the computational effort, but also is helpful for obtaining better component values. However, as the PCS part is always with static characteristics and the component values are relative stable, the components in PCS are not optimized in this paper, but the components in the FN part which are crucial to the circuit performance are optimized by OLPSO.

B. Orthogonal Learning Particle Swarm Optimization

PSO is introduced as a swarm intelligence (SI) [28] algorithm that emulates the swarm behaviors of birds flocking and fish schooling. When search in a D-dimensional space, each particle i keeps a velocity vector $\boldsymbol{V_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ and a position vector $\boldsymbol{X_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ to indicate its current state, where i is a positive integer indexing the particle in the swarm and D is the dimensionality of the problem. Moreover, the personal historically best position vector of particle i is denoted as $\boldsymbol{P_i} = [p_{i1}, p_{i2}, \ldots, p_{iD}]$. The best position of all the particles in the i^{th} particle's neighborhood (the neighborhood of a particle is defined by a topology structure, e.g., the neighborhood of particle i includes the particles i–1, i, and i+1 in a ring topology structure) is denoted as $\boldsymbol{P_n} = [p_{n1}, p_{n2}, \ldots, p_{nD}]$. The vectors $\boldsymbol{V_i}$ and $\boldsymbol{X_i}$ are initialized randomly and are updated by Eqs. (1) and (2) generation by generation through the guidance of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$.

$$v_{id} = v_{id} + c_1 r_{1d}(p_{id} - x_{id}) + c_2 r_{2d}(p_{nd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

The ω is inertia weight introduced by Shi and Eberhart [29] to control the exploration and exploitation abilities of the algorithm. Parameters c_1 and c_2 are acceleration coefficients which are commonly set to 2.0. The r_{1d} and r_{2d} are two randomly generated values within range [0, 1] for the d^{th} dimension. In order to control the flying velocity within a reasonable range, a positive value $V_{\text{MAX}d}$ (can be set to 20% of the search range of the corresponding dimension [30]) is used to clamp the updated velocity. If $|v_{id}|$ exceeds $V_{\text{MAX}d}$, then it is set to $sign(v_{id})V_{\text{MAX}d}$. Similarly, if the updated position x_{id} particle is out of the search range $[X_{\text{MIN}d}, X_{\text{MAX}d}]$, the position is set to the corresponding bound. The new position is evaluated and can replace the $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$ if it has a better fitness value. Then the algorithm goes on for the next generation until termination.

The difference between OLPSO and traditional PSO is that OLPSO use an OL strategy to combine the information of $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ to form a better guidance vector $\boldsymbol{P_o}$. Then the particle's flying velocity is thus adjusted as:

$$v_{id} = \omega v_{id} + c r_d(p_{od} - x_{id}) \tag{3}$$

where ω is the same as in (1) and c is fixed to 2.0, the same as c_1 and c_2 in (1), and r_d is a random value generated within the range [0, 1] for the d^{th} dimension.

Each particle i has its own guidance vector $\boldsymbol{P_o}$, which is denoted as $\boldsymbol{P_o} = [p_{o1}, p_{o2}, \ldots, p_{oD}]$ and is constructed from $\boldsymbol{P_i}$ and $\boldsymbol{P_n}$ as

$$\boldsymbol{P_o} = \boldsymbol{P_i} \oplus \boldsymbol{P_n} \tag{4}$$

where the symbol $\oplus$ stands for the OED operation.

OED is one of the most important experimental design methods and has been used to solve many classes of real-world problems successfully [26][27]. The OED method has been developed for the purpose of tracking experimental problems with many kinds of *factors* and many different choices per factor, namely *levels*. The efficiency of OED lies in its ability of a *general-design* test by using an *orthogonal array* (OA) to construct representative combinations, and its ability of using a *factor analysis* (FA) to find out or predict the best combination by testing only a limited number of cases. The OA for a problem with N factors and Q levels per factor is always denoted by $L_M(Q^N)$, where L denotes the OA and M is the number of combinations of test cases.

In OLPSO, the OED is based on a two levels multiple factors OA because each dimension is regarded as a factor and the choice from P_{id} or P_{nd} of the the d^{th} dimension are regarded as two levels. When using OLPSO to optimize a problem with D dimensions, the construction process of $\boldsymbol{P_o}$ based on the OED (refer to Eq. (4)) is described as the following six steps.

Step 1: An OA is generated as $L_M(2^D)$, using the approach detailed in [21][27], where D is the number of dimensions (factors) and $M = 2^{\lceil \log_2(D+1) \rceil}$. Table I is an example using an OA with 7 factors and each factor with 2 levels.

TABLE I. AN EXAMPLE OF ORTHOGONAL EXPERIMENTAL DESIGN WITH SEVEN FACTORS AND TWO LEVELS

OED	Factors	A	B	C	D	E	F	G	Fitness
Combinations According to OA	$\boldsymbol{X_1}$	1	1	1	1	1	1	1	f_1
	$\boldsymbol{X_2}$	1	1	1	2	2	2	2	f_2
	$\boldsymbol{X_3}$	1	2	2	1	1	2	2	f_3
	$\boldsymbol{X_4}$	1	2	2	2	2	1	1	f_4
	$\boldsymbol{X_5}$	2	1	2	1	2	1	2	$\boldsymbol{f_5}$ (max)
	$\boldsymbol{X_6}$	2	1	2	2	1	2	1	f_6
	$\boldsymbol{X_7}$	2	2	1	1	2	2	1	f_7
	$\boldsymbol{X_8}$	2	2	1	2	1	1	2	f_8
FA for Contributions	L_1	$\boldsymbol{S_{A1}}$	S_{B1}	S_{C1}	S_{D1}	$\boldsymbol{S_{E1}}$	$\boldsymbol{S_{F1}}$	S_{G1}	
	L_2	S_{A2}	$\boldsymbol{S_{B2}}$	$\boldsymbol{S_{C2}}$	$\boldsymbol{S_{D2}}$	S_{E2}	S_{F2}	$\boldsymbol{S_{G2}}$	
Best	$\boldsymbol{X_b}$	L_1	L_2	L_2	L_2	L_1	L_1	L_2	f_b
Make up $\boldsymbol{P_o}$	$\boldsymbol{P_o}$	L_2	L_1	L_2	L_1	L_2	L_1	L_2	if $f_5 > f_b$
		L_1	L_2	L_2	L_2	L_1	L_1	L_2	if $f_b > f_5$

Step 2: Make up M tested solutions $\boldsymbol{X_j}$ ($1 \le j \le M$) by selecting the corresponding value from $\boldsymbol{P_i}$ or $\boldsymbol{P_n}$. Here, if the value in the OA is 1, then the corresponding factor (dimension) selects $\boldsymbol{P_i}$; otherwise, selects $\boldsymbol{P_n}$.

Step 3: Evaluate each tested solution $\boldsymbol{X_j}$ ($1 \le j \le M$), and record the best solution $\boldsymbol{X_b}$. The fitness of each combination is listed in the last column of Table I. For example, the best combination is $\boldsymbol{X_5}$ for this maximization problem.

Step 4: Calculate the effect of each level on each factor and determine the best level for each factor using the FA process.

The FA is to calculate the contributions of different levels on each factor. The process works as follows. Let f_m denote the experimental result of the m^{th} ($1 \le m \le M$) combination and S_{dq} denote the effect of the q^{th} ($1 \le q \le Q$) level in the d^{th} ($1 \le d \le D$) factor. The calculation of S_{dq} is to add up all the f_m in which the level is q in the d^{th} factor, and then divide the total count of z_{mdq}, as shown in (5) where z_{mdq} is 1 if f_m is the q^{th} level in the d^{th} factor, otherwise, z_{mdq} is 0.

$$S_{dq} = \frac{\sum_{m=1}^{M} f_m \times z_{mdq}}{\sum_{m=1}^{M} z_{mdq}} \tag{5}$$

In this way, the effect of each level on each factor can be calculated and compared, as shown in Table I. For example, when we calculate the effect of level 1 on factor A, denoted by element A1, the experimental results of $\boldsymbol{X_1}$, $\boldsymbol{X_2}$, $\boldsymbol{X_3}$, and $\boldsymbol{X_4}$ are summed up for Eq. (5) because only these combinations are involved in level 1 of factor A. Then the sum divides the combination number (4 in this case) to yield S_{dq} (S_{A1} in this case). With all the S_{dq} calculated, the best combination of the levels can be determined by selecting the level of each factor that provides the highest-quality S_{dq}. For example, for a maximization problem, if $S_{\text{A1}} > S_{\text{A2}}$, then the better level of factor A is level 1.

Step 5: Derive a predictive solution $\boldsymbol{X_p}$ with the levels determined in Step 4 and evaluate $\boldsymbol{X_p}$.

Step 6: Compare $f(\boldsymbol{X_b})$ and $f(\boldsymbol{X_p})$ and the level combination of the better solution is used to construct the vector $\boldsymbol{P_o}$. In the example of Table I, f_5 is compared with $f(\boldsymbol{X_p})$, if f_5 is better than $f(\boldsymbol{X_p})$, then $\boldsymbol{P_o}$ is constructed with the level combination of $\boldsymbol{X_5}$; otherwise, $\boldsymbol{P_o}$ is constructed with the level combination of $\boldsymbol{X_p}$.

III. OLPSO FOR PEC

A. Particle Representation

Using OLPSO to solve the PEC, the components in the FN part can be represented with the use of an vectors $X(FN)$. Specifically, the representation of each particle for optimizing the FN components is coded as

$$X(FN) = [\overline{R}_F \quad \overline{I}_F \quad \overline{C}_F] \tag{6}$$

where $\overline{R}_F = [R_1, R_2, \cdots, R_{R_F}]$ are the resistors, $\overline{I}_F = [I_1, I_2, \cdots, I_{I_F}]$ are the inductors, and $\overline{C}_F = [C_1, C_2, \cdots, C_{C_F}]$ are the capacitors

B. Fitness Function

The fitness function definition for FN is according to the proposals in [16] whose main considerations includes reducing the settling time and controlling the overshoot. The fitness function is described as:

$$\Phi_{FN}(X) = \sum_{R_L = R_{L\_\min}, \delta R_L}^{R_{L\_\max}} \sum_{v_{in} = v_{in\_\min}, \delta v_{in}}^{v_{in\_\max}} [F_1(R_L, v_{in}, X) + F_2(R_L, v_{in}, X) + F_3(R_L, v_{in}, X)] + F_4(X) \tag{7}$$

where $R_{L\_\min}$ and $R_{L\_\max}$, $v_{in\_\min}$ and $v_{in\_\max}$ are the minimal and maximal values of R_L and v_{in}, respectively. δR_L and δv_{in} are the step length in varying the values of R_L and v_{in}.

The F_1, F_2, F_3, and F_4 are the four objective functions for the FN as designed in [16]. Specifically, F_1 is to measure the steady-state error of the output voltage v_o; F_2 is to measure the transient response of v_d, including the maximum overshoot and undershoot, and the settling time; F_3 is to control the steady-state ripple voltage on the output v_o; F_4 is to measure the dynamic behaviors during the large-signal change. For more details of the fitness function definitions, refer to [16].

C. Implementation of OLPSO for PEC

With the particle representation and fitness function defined as above, the implementation of using OLPSO to search the optimal component values for the PEC is described as the following four steps.

Step 1: Initialization. For each particle i, initialize its position $\boldsymbol{X_i}$ (the circuit components) and its velocity $\boldsymbol{V_i}$ with random values in their corresponding search ranges $[X_{\text{MIN}d}, X_{\text{MAX}d}]$ and $[-V_{\text{MAX}d}, V_{\text{MAX}d}]$, respectively. Set its personal best position $\boldsymbol{P_i}$ as $\boldsymbol{X_i}$. Calculate all the particles' fitness and determine the neighborhood best $\boldsymbol{P_n}$ for each particle. Determine the global best particle as $\boldsymbol{gBest} = \max\{fit(\boldsymbol{P_i})\}$. Construct the orthogonal learning example $\boldsymbol{P_o}$ via OL strategy (Eq. (4)) for each particle. Set $gen=0$, $stop_i=0$.

Step 2: Evolutionary process. For each particle i:

Step 2.1: If $stop_i > G$, reconstruct the orthogonal learning example $\boldsymbol{P_o}$ via OL strategy (Eq. (4)) for particle i. Set $stop_i=0$.

Step 2.2: Update the velocity $\boldsymbol{V_i}$ as Eq. (3), and clamp the velocity in the range $[-V_{\text{MAX}d}, V_{\text{MAX}d}]$.

Step 2.3: Update the position $\boldsymbol{X_i}$ as Eq. (2), and clamp the position in the search range $[X_{\text{MIN}d}, X_{\text{MAX}d}]$.

Step 2.4: Evaluate the new position $\boldsymbol{X_i}$. If $fit(\boldsymbol{X_i}) > fit(\boldsymbol{P_i})$, then set $\boldsymbol{P_i} = \boldsymbol{X_i}$, and set $stop_i=0$. Otherwise, set $stop_i = stop_i+1$.

Step 2.5: If $fit(\boldsymbol{P_i}) > fit(\boldsymbol{P_n})$, then set $\boldsymbol{P_n} = \boldsymbol{P_i}$. If $fit(\boldsymbol{P_i}) > fit(\boldsymbol{gBest})$, then set $\boldsymbol{gBest} = \boldsymbol{P_i}$.

Step 2.6: If all the particles have finished Step 2.1 to Step 2.6, go to Step 3. Otherwise, go to Step 2.1 for the next particle.

Step 3: Termination check. If the number of iterations or the number of fitness evaluations reaches the maximum, then go to Step 4. Otherwise, set $gen=gen+1$, and go to Step 2 for the next generation evolution.

Step 4: Print the global best particle $\boldsymbol{gBest}$ as the final solution.

IV. DESIGN EXAMPLE AND RESULTS COMPARISONS

A. Circuit Configurations

In this section, the OLPSO algorithm is applied to the PEC design and optimization problem. A buck regulator with overcurrent protection as shown in Fig. 2 is used as simulation

case [16][17]. In this circuit, the PCS part is a classical buck converter and the FN part is a proportional-plus-integral controller. For PCS, only the L and C are required to be optimized whilst the R_L, r_C, and r_E are assumed to be known as in a practical circuit. Moreover, the PCS part is always with static characteristics and the components L and C are relatively stable [16]. Therefore, the components in PCS are not optimized in this paper, but the values are set as 200μH and 1000μF for L and C, respectively, according the proposals in [16] and by the considerations of available component values in industry. For FN, all component values are required to be optimized. That is, the components R_1, R_2, R_{C3}, R_4, C_2, C_3, and C_4 in the FN part are optimized by OLPSO and the fitness function is as (7). Moreover, the required specifications of the whole PEC are listed as follows:

1) Input voltage range v_{in}: 20 ~ 40 V

2) Output load range R_L: 5 ~ 10 Ω

3) Nominal output voltage: 5 V±1%

4) Switching frequency: 20 kHz

5) Maximum settling time: 20 ms

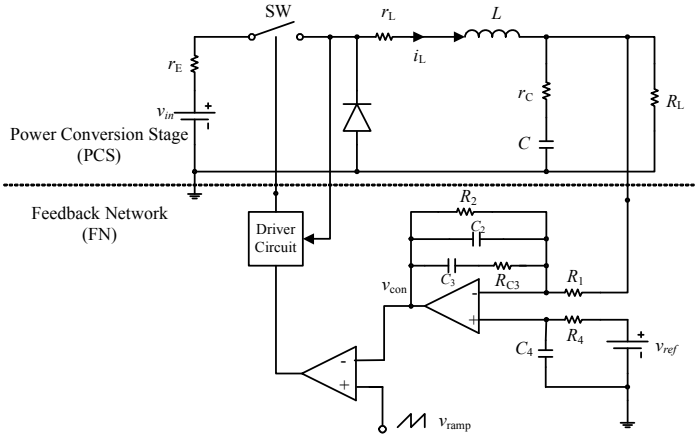

Figure 2. Circuit schematics of the buck regulator with overcurrent protection.

In the previous studies [16]-[19], the components' search ranges are carefully pre-defined by expert designers in order to make the optimization approaches easily to search. However, such specific search ranges are difficult to be defined for different PECs. In this paper, we set the components search ranges freely according to commonly used ranges. That is, the search range for resistors R_1, R_2, R_{C3}, and R_4 are all set to be [100Ω, 100kΩ] and the search range for capacitors C_2, C_3, and C_4 are all set to be [0.1μF, 100μF], as given in Table II.

TABLE II. SEARCH RANGES OF THE COMPONENTS IN THE FN PART

Components	Search Ranges
R_1	[100Ω, 100kΩ]
R_2	[100Ω, 100kΩ]
R_{C3}	[100Ω, 100kΩ]
R_4	[100Ω, 100kΩ]
C_2	[0.1μF, 100μF]
C_3	[0.1μF, 100μF]
C_4	[0.1μF, 100μF]

B. Algorithm Configurations

The performance of OLPSO in optimizing the PEC is evaluated and compared with both the GA approach proposed in [16] and the PSO approach in [23]. The parameters of GA and PSO are set according to the configurations in their references. For the population size and the maximal generation number, they are set to be 30 and 500 respectively in both GA and PSO as proposed in [16] and [23], respectively. The population size OLPSO is set to be 40 according to the proposal in [21]. However, in order to make a fair comparison, all the algorithms use the same maximal fitness evaluations (FEs) of 1.5×10^4 (the value of 30×500) as the termination criterion [16][23]. As the evaluation of the fitness function is usually the most expensive computational part in the optimization of PEC, the execution time of different algorithms will be almost the same if they use the same number of FEs.

The crossover and mutation probabilities of the GA approach are set the same as in [16] where p_x=0.85 and p_m=0.25. The inertia weight ω in PSO and OLPSO linearly decreases from 0.9 to 0.4, while the acceleration coefficients c_1 and c_2 in PSO and the acceleration coefficient c in OLPSO are all set to be 2.0. Moreover, is also 2.0 and the parameter G is set to be 5 [20][21]. In order to make the comparisons in a statistical sense, the experiment is carried out 30 times independently with each approach and the average results are used for comparison.

C. Comparisons on Fitness Quality

The results of GA, PSO, and OLPSO are compared in Table III where the "Mean" stands for the average fitness value of the 30 independent runs and "Std. Dev" is the standard deviation. Moreover, the "Best" fitness values among the 30 runs are given and compared in the table. It can be observed from the table that OLPSO achieves better results than both GA and PSO when measured by the mean fitness value. Therefore, OLPSO is capacity to obtain good solutions consistently. The t-Test further confirms that OLPSO outperforms PSO and GA significantly and the results obtained by OLPSO are remarkably better. Moreover, OLPSO can obtain the highest "Best" fitness solution among the three approaches, indicating the strong global search ability of OLPSO. The obtained component values in the "Best" fitness solution optimized by different approaches are presented in Table IV.

TABLE III. RESULT COMPARISONS OF DIFFERENT APPROACHES

Approaches	Mean	Std. Dev	t-Test	Best	Worst	errFEs	Success #
GA	109.636	9.00583	-24.15414†	127.494	97.1908	×	0
PSO	152.11	21.29	-6.77323‡	192.304	137.699	3817	8
OLPSO	**183.748**	**14.1892**	NA	**192.962**	**137.924**	**3675**	**28**

†The difference between GA and OLPSO is significant with 29 degrees of freedom at α=0.05 by a two-tailed t-test.
‡The difference between PSO and OLPSO is significant with 29 degrees of freedom at α=0.05 by a two-tailed t-test.
The 'errFEs' is the mean fitness evaluations to reach an acceptable fitness of 150 among all the successful runs.
The 'Success #' is the count of runs that can find final solution with a fitness larger than 150.

It is should be reminded that these results are obtained in the new configured large search space. The total failure of GA indicates that this approach is not efficient enough to make sufficient search in the large space to find the good solution, even though it is promising in the carefully pre-defined search range [16]. Moreover, the large search space challenges the search ability of traditional PSO algorithm. The OLPSO

algorithm is still promising and its results are demonstrated to be much better than the others.

TABLE IV. OPTIMIZED COMPONENT VALUES IN THE BEST RUN WITH DIFFERENT APPROACHES

Components	GA	PSO	OLPSO
R_1	356.099 Ω	100 Ω	100 Ω
R_2	60.4418 kΩ	71.7442 kΩ	13.1202 kΩ
R_{C3}	98.6189 kΩ	831.532 Ω	1.04713 kΩ
R_4	2.07867 kΩ	11.4945 kΩ	11.1206 kΩ
C_2	19.6276 μF	0.1 μF	0.1 μF
C_3	28.0941 μF	1.72671 μF	1.11032 μF
C_4	3.38356 μF	0.1 μF	0.1 μF
Fitness Value	127.494	192.304	**192.962**

D. Comparisons on Optimization Speed

Besides the high solution quality of OLPSO, the fast optimization speed and strong algorithm reliability of OLPSO are also supported by the comparisons in Table III. By giving an acceptable fitness value of 150, OLPSO can successfully obtain final solutions with fitness values larger than 150 in 28 out of the 30 runs whilst PSO can only succeeds in 8 runs and GA totally fails in obtaining solutions with fitness values larger than 150. Moreover, among the successful runs in each algorithm, the mean FEs needed to reach the acceptable value of 150 in Table III further shows that OLPSO is the fastest algorithm among the three contenders.

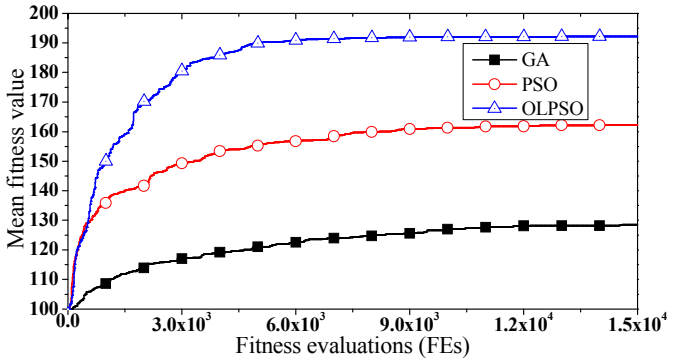

Figure 3. Mean convergence characteristics of different approaches in optimizing the FN.

The mean convergence characteristics of different approaches are plotted in Fig. 3. The curves show that both GA and PSO fall into a poor local optimum quite early whilst OLPSO is able to obtain very high fitness in early state and to improve the fitness value steadily for a long time. OLPSO has strong global search ability to avoid local optima and has significantly improved the fitness value. Moreover, the curves indicate that OLPSO is faster than the other algorithms to optimize the component values. That is, when a fixed fitness is given, OLPSO is observed to use much less FEs to obtain this specific value than the GA or PSO algorithm.

E. Comparisons on Simulation Results

In the simulations, the component values of the PEC are set as the best optimized results obtained by GA and OLPSO respectively, as given in Table IV. In the simulation results comparison, Fig. 4 gives the results of voltage and Fig. 5 gives the results of current. We do not give the simulation results of the circuit obtained by PSO because the best solution of PSO is similar to that of OLPSO and the simulation results of the PSO optimized best circuit are similar to that of the best circuit obtained by OLPSO. However, the OLPSO outperforms PSO significantly in mean solution quality, optimization speed, and algorithm reliability, as demonstrated in the above sub-section.

The simulation lasts for 90 milliseconds (ms). The input voltage v_{in} is 20 V and the output load R_L is 5 Ω. The simulated startup transients can be compared in the first 30 ms of the figures. It is observed that the circuit with OLPSO-optimized component values has better performance, giving faster settling time. The buck with component values optimized by OLPSO uses only about 5 ms to reach the steady state, while the one with component values optimized by GA uses about 10 ms. Moreover, the output ripple voltage of the OLPSO-optimized circuit is less than 1%, satisfying the required specification very well.

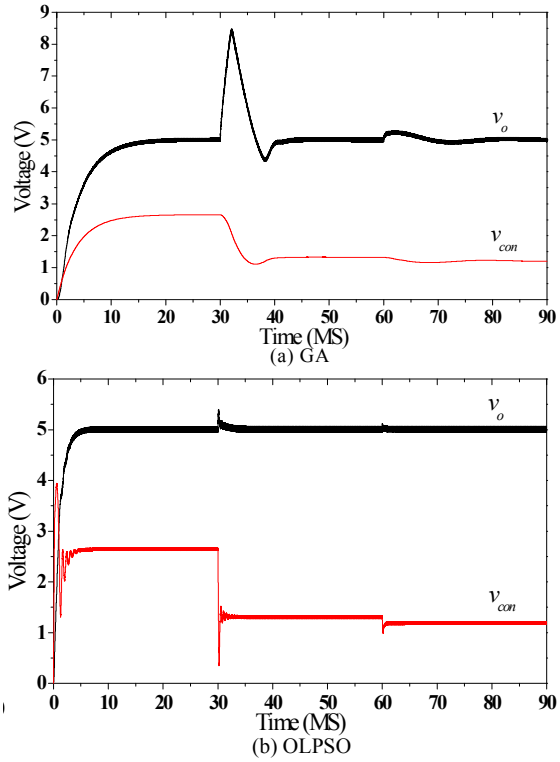

Figure 4. Simulated voltage responses from 0 ms to 90 ms. From 0 ms to 30 ms, v_{in} is 20 V and R_L is 5 Ω; on the 30 ms, v_{in} is suddenly changed from 20 V to 40 V while R_L is still 5 Ω; on the 60 ms, R_L is suddenly changed from 5 Ω to 10 Ω while v_{in} is still 40 V. (a) Simulated voltage responses of the regulator with component values optimized by GA. (b) Simulated voltage responses of the regulator with component values optimized by OLPSO.

Fig. 4 and Fig. 5 also show the simulated transient responses under large signal disturbances. On the 30 ms, when the regulator is in steady state, the input voltage is suddenly changed from 20 V to 40 V, with the load still fixed as 5 Ω. As the responses to this change, the output voltage v_o, the control voltage v_{con}, and the inductor current i_L are all disturbed. However, the circuit optimized by OLPSO has much smaller disturbance and shorter response time than the one optimized by GA (2ms vs 10ms), confirming the advantages of the OLPSO algorithm.

Similar tests on load disturbances are also studied when the system has reverted a steady state with v_{in} equals 40 V and R_L equals 5 Ω. In this disturbance, R_L is suddenly changed from 5 Ω to 10 Ω on the 60 ms, with the v_{in} is still fixed as 40 V. The simulation results in the figures also show that the OLPSO-optimized circuit has a smaller disturbance response to the change and a shorter time to revert the steady state. Therefore, the OLPSO algorithm can optimize the circuit component values to make the circuit exhibit better dynamic performance.

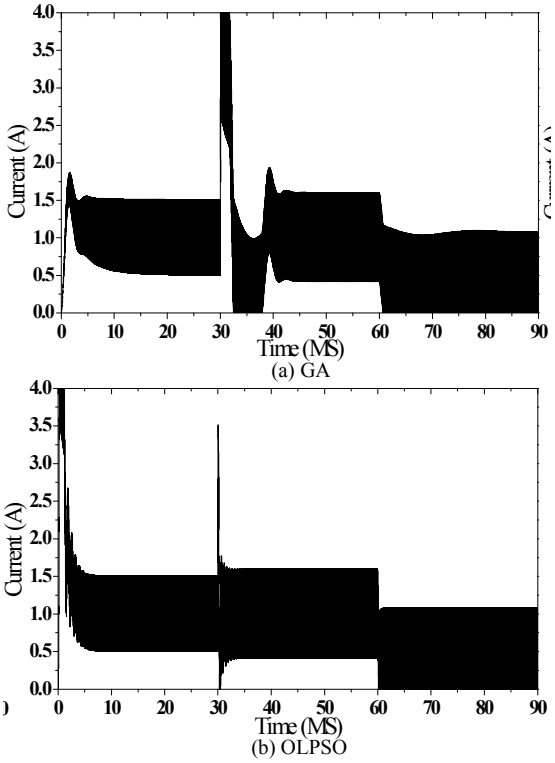

Figure 5. Simulated voltage responses from 0 ms to 90 ms. From 0 ms to 30 ms, v_{in} is 20 V and R_L is 5 Ω; on the 30 ms, v_{in} is suddenly changed from 20 V to 40 V while R_L is still 5 Ω; on the 60 ms, R_L is suddenly changed from 5 Ω to 10 Ω while v_{in} is still 40 V. (a) Simulated voltage responses of the regulator with component values optimized by GA. (b) Simulated voltage responses of the regulator with component values optimized by OLPSO.

F. Comparisons on Discrete Search Space

The comparisons on fitness quality, optimization speed, and simulation results have demonstrated that OLPSO performs much better than both the GA approach in [16] and the PSO approach in [23]. However, all these results are based on the continuous search space and therefore the optimized component values are sometimes not readily available in manufactory, but require post-fabrication. For example, the results in Table IV show that the optimized value for the resistor R_2 is 13.1202 kΩ and the optimized value for the capacitor C_3 is 1.11032 μF. These values are needed to be composed by connecting several resistors or capacitors in series and/or parallel, and therefore make it not easy to use in practical applications. As the resistors and capacitors are always manufactured with discrete values [31], we will test the search abilities of different algorithms in obtaining the optimized component values in the discrete search space.

In the following experiments in this sub-section, the optimization algorithms work the same as they are searching in a continuous search space except that when evaluating the fitness value of an individual, the variables are first rounded to the nearest readily available value. For example, in the E24 series of resistor and capacitor, the resistor values are among {…, 180Ω, 200Ω, 220Ω, …, 12kΩ, 13kΩ, 15kΩ, …} and the capacitor values are among {…, 0.33μF, 0.36μF, 0.39μF, …, 1.1μF, 1.2μF, 1.3μF, …} [31]. Therefore, if the algorithm finds a resistor value of 13.1202kΩ, it will be rounded to 13kΩ, and if the capacitor value is 1.11032μF, it will be rounded to 1.1μF.

We apply such a strategy to the GA, PSO, and OLPSO algorithms to test their search abilities in discrete space whose ranges are still the same as given in Table II. The experimental results are compared in Table V and the obtained component values in the "Best" fitness solution optimized by different approaches are presented in Table VI. The results show that GA still totally fails in the discrete search space, PSO only successes less than 50% (13 times out of 30 trials), and OLPSO successes 25 times. The mean fitness obtained by OLPSO is much better than both GA and PSO, as indicated by the *t*-Test. Therefore, OLPSO has advantages in optimizing the PEC not only when the search space is continuous, but also when the search space is discrete.

TABLE V. RESULT COMPARISONS ON DISCRETE SEARCH SPACE

Approaches	Mean	Std. Dev	*t*-Test	Best	Worst	errFEs	Success #
GA	104.228	7.04873	-19.07939†	128.089	96.9369	×	0
PSO	156.287	26.856	-3.34232‡	191.842	111.801	**2063**	13
OLPSO	**176.55**	**19.5284**	NA	**192.199**	**134.021**	4072	**25**

†The difference between GA and OLPSO is significant with 29 degrees of freedom at α=0.05 by a two-tailed *t*-test.
‡The difference between PSO and OLPSO is significant with 29 degrees of freedom at α=0.05 by a two-tailed *t*-test.
The 'errFEs' is the mean fitness evaluations to reach an acceptable fitness of 150 among all the successful runs.
The 'Success #' is the count of runs that can find final solution with a fitness larger than 150.

TABLE VI. OPTIMIZED COMPONENT VALUES IN THE BEST RUN WITH DIFFERENT APPROACHES IN DISCRETE SEARCH SPACE

Components	GA	PSO	OLPSO
R_1	200 Ω	100 Ω	100 Ω
R_2	91 kΩ	30 kΩ	10 kΩ
R_{C3}	100 kΩ	620 Ω	1.1 kΩ
R_4	4.7 kΩ	100 Ω	100 Ω
C_2	43 μF	0.1 μF	0.1 μF
C_3	36 μF	2.2 μF	0.91 μF
C_4	1.8 μF	12 μF	11 μF
Fitness Value	128.089	191.842	**192.199**

V. CONCLUSIONS

This paper presents an orthogonal learning PSO for optimizing the component values in designing PEC. The challenge of the problem is that the components interact with each other and make the search space complex. Moreover, this paper argues that the search range of the components is not easy to be determined within certain specific interval, and proposes to set the free search range for the components according to the commonly used range. To optimize PEC with large and complex search space, the distinct feature of the proposed OLPSO algorithm is that it uses a novel orthogonal learning strategy that can discover useful information in a particle's personal best position and its neighborhood's best position. This new OL learning strategy helps a particle construct a more promising and efficient guidance exemplar to

improve its flying velocity and direction, and therefore is promising in a large search space.

The effectiveness and efficiency of the OLPSO algorithm in optimally designing PEC have been evaluated with the design of a buck regulator with overcurrent protection. In order to demonstrate the advantages of the proposed OLPSO algorithm, results obtained by GA and also by PSO with traditional learning strategy are compared with the ones obtained by OLPSO. The results show that OLPSO outperforms the other algorithms not only with higher quality fitness value, but also with faster optimization speed and stronger algorithm reliability. Moreover, simulations on the circuits optimized by OLPSO and GA are conducted and compared. The simulation results demonstrate the advantages of the OLPSO algorithm by showing that the circuit optimized by OLPSO exhibits both shorter startup time and short settling time in the transient responses. The disturbance of output voltage and current in the OLPSO-optimized circuit is also slighter than that in the GA-optimized circuit. The experiments are also conducted in the discrete search space. The results show that OLPSO still has superior performance.

REFERENCES

[1] B. K. Bose, "Energy, environment, and advances in power electronics," *IEEE Trans. Power Electron.*, vol. 15, no. 4, pp. 688-701, Jul. 2000.

[2] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics: Converters, Applications and Design*. New York: Wiley, 2001.

[3] J. G. Kassakian, M. F. Schlecht, and G. C. Verghese, *Principles of Power Electronics*, Addison-Wesley, 1991.

[4] A. Emadi, "Modeling of power electronic loads in AC distribution systems using the generalized state-space averaging method," *IEEE Trans. Ind. Electron.*, vol. 51, no. 5, pp. 992-1000, Oct. 2004.

[5] P. R. K. Chetty, "Current injected equivalent circuit approach to modeling of switching dc-dc converters in discontinuous inductor conduction mode," *IEEE Trans. Ind. Electron.*, vol. IE-29, no. 3, pp. 230-234, Aug. 1982.

[6] G. C. Vergliese, M. E. Elbuluk, and J. G. Iiassakian, "A general approach to sampled-data modeling for power electronic circuits," *IEEE Trans. Power Electronics*, vol. PE-1, pp. 76-89, Apr. 1986.

[7] R. Oruganti and F. C. Lee, "State-plane analysis of parallel resonant converter," *IEEE PESC Record*, 1985, pp. 56-73.

[8] G. J. Sussman and R. M. Stallman, "Heuristic techniques in computer aided circuit analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-22, no. 11, pp. 857–865, Nov. 1975.

[9] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Trans. Comput.-Aided Design*, vol. 8, no. 6, pp. 1247–1266, Jun. 1989.

[10] L. P. Huelsman, "Optimization—A powerful tool for analysis and design," *IEEE Trans. Circuits Systs. I, Reg. Papers*, vol. 40, no. 7, pp. 431–439, Jul. 1993.

[11] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX," *IEEE Trans. Comput.-Aided Design*, vol. 15, no. 3, pp. 273–294, Mar. 1996.

[12] Z. H. Zhan, J. Zhang, Y. Li, O. Liu, S. K. Kwok, W. H. Ip, and O. Kaynak, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. on Intell. Transport. Syst.*, vol. 11, no. 2, pp. 399-412, Jun. 2010.

[13] Z. H. Zhan, X. L. Feng, Y. J. Gong, and J. Zhang, "Solving the flight frequency programming problem with particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, May. 2009, pp. 1383-1390.

[14] Z. H. Zhan, J. Zhang, and Zhun Fan, "Solving the optimal coverage problem in wireless sensor networks using evolutionary computation algorithms," *The 8th International Conference on Simulated Evolution And Learning, SEAL2010, LNCS 6457*, pp. 166−176, 2010.

[15] Z. H. Zhan and J. Zhang, "Discrete particle swarm optimization for multiple destination routing problems," *EvoWorkshops 2009*, *LNCS 5484*, 2009, pp. 117−122.

[16] J. Zhang, H. Chung, W. L. Lo, S. Y. R. Hui, and A. Wu, "Implementation of a decoupled optimization technique for design of switching regulators using genetic algorithm," *IEEE Trans. Power Electron.*, vol. 16, no. 6, pp. 752–763, Nov. 2001.

[17] J. Zhang, S. H. Chung, W. L. Lo, and T. Huang, "Extended ant colony optimization algorithm for power electronic circuit design," *IEEE Trans. Power Electron.*, vol. 24, no. 1, pp. 147-162, Jan. 2009.

[18] J. Zhang, W. L. Lo, and S. H. Chung, "Pseudocoevolutionary genetic algorithms for power electronic circuits optimization," *IEEE Trans. Syst., Man, and Cybern.,C*, vol. 36, no. 4, pp. 590-598, Jul. 2006.

[19] J. Zhang, S. H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 326-335, Jun. 2007.

[20] Z. H. Zhan, J. Zhang, and O. Liu, "Orthogonal learning particle swarm optimization," in *Proc. Genetic Evol. Comput. Conf.*, Montréal, Canada, Jul., 2009, pp. 1763-1764.

[21] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, (in press).

[22] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, Perth, Australia, vol. 4, 1995, pp. 1942-1948.

[23] J. Zhang, Y. Shi, and Z. H. Zhan, "Power electronic circuits design: A particle swarm optimization approach," *The 7th International Conference on Simulated Evolution And Learning, X. Li et al. (Eds.): SEAL 2008*, LNCS 5361, pp. 605–614. 2008.

[24] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 211-224, Jun. 2004.

[25] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp.225-239, Jun. 2004.

[26] D. C. Montgomery, *Design and Analysis of Experiments*, 5th ed. New York: Wiley, 2000.

[27] Math. Stat. Res. Group, Chinese Acad. Sci., *Orthogonal Design* (in Chinese). Beijing: People Education Pub., 1975.

[28] J. Kennedy, R. C. Eberhart, and Y. H. Shi, *Swarm Intelligence*, San Mateo, CA: Morgan Kaufmann, 2001.

[29] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69-73.

[30] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, and Cybern. B.*, vol. 39, no. 6, pp. 1362-1381, Dec. 2009.

[31] Electronics 2000, http://www.electronics2000.co.uk/, access date: April. 2011.

[32] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.