Optimization of Power Electronic Circuits Using Ant Colony System

Jun Zhang[*], *Member, IEEE*, Henry S.H. Chung[**], *Senior Member, IEEE*, Alan W. L. Lo[***] and Tao Huang [*]

[*]Dept. of Computer Science, Sun-Yat-Sen University, Guangzhou, P. R. China
[**]Center for Power Electronics, City University of Hong Kong, Hong Kong
[***]Dept. of Computer Science, Chu Hai College of Higher Education, Hong Kong

***Abstract*—Ant colony optimization (ACO) is typically used to search paths through graphs. The concept is based on simulating the behavior of ants in finding paths from the colony to food. Its searching mechanism is applicable for optimizing electric circuits with components, like resistors and capacitors, available in discrete values. In this paper, an extended ACO (eACO) that can search the optimal values of components manufactured in discrete and continuous values is presented. The idea is based on using the orthogonal design method (ODM) to dynamically update the database of the continuous components so that those components will have *pseudo*-discrete values in the search space. To speed up the optimization process, the ODM performs local search of the best combination around the best ant. The eACO also takes the component tolerances into account in evaluating the fitness value of each ant. The proposed algorithm has been successfully used to optimize the design of a buck regulator. The predicted results have been compared with the published results available in the literature and have also verified with experimental measurements.**

I. INTRODUCTION

Small-signal techniques have been dominantly used to model, design and analyze a power electronics circuit (PEC) [2]-[5]. The procedures are simple, but they require detailed knowledge on the circuit operations, such as operating modes and control schemes. Various proposals for analog circuit design automation have been emerged in the early 1970's. Those methods incorporate heuristics [6], knowledge bases [7], simulated annealing [8], and other algorithms. Classical optimization techniques such as the gradient methods and hill-climbing techniques have been applied [9]-[10]. However, they might be subject to becoming trapped into local minima, leading to sub-optimal parameter values, and thus, having a limitation of operating in large, multimodal, and noisy spaces.

Genetic algorithm (GA) [11], which is one of the *meta-heuristic* optimization methods, has been shown to be an effective way to find solutions close to the global optimum. It is less dependent upon the initial guess and has been applied to optimize specific parts in the power electronic systems, such as system controllers [12]-[17], modulation schemes [18]-[19], optimization of component values [20]-[22], and specific applications, like battery chargers in [23]-[24]. However, the values optimized by GAs for the circuits are sometimes not available readily, but require post-fabrication to to compose the values optimized by GAs. Moreover, inevitable component tolerance would make the actual component value differ from its nominal values to some extent.

Recently, a new *meta-heuristic* optimization method named ant colony optimization (ACO) has been proposed in [25]-[29]. The method simulates the behaviors of ants in finding paths from the colony to food. It is a multi-agent approach for solving combinatorial optimization problems, like traveling salesman problems [26], data mining [29], network routing [30], and controller design [31]. Recently, there is a growing application of ACO to electrical engineering, such as power distribution system planning and load flow study [32]-[34]. Exploration of ACO to PEC design and optimization is progressed at a slow pace. The search mechanism in the ACO can be applicable for optimizing circuits with components, like resistors and capacitors, available in discrete values. As illustrated in Fig. 1, the ACO is based on using several ants to search the best combination among various possible combinations of the resistor and capacitor values, such as the values of R_1, R_2, …, in $\overline{R}_F$, and C_1, C_2, …, in $\overline{C}_F$ in the feedback network shown in Fig. 2. However, PEC generally consists of components, such as the values of the inductors L_1, L_2, …, in $\overline{L}_F$, and the parameters, such as the voltage gains g and H_o in Fig. 2, which are designed with continuous values. Thus, the traditional ACO cannot be applied.

In this paper, an extended ACO (eACO) that can search the optimal values of components manufactured in discrete and continuous values is presented. The idea is based on using the orthogonal design method (ODM) to dynamically update the database of the components manufactured with continuous values, so that those components will have *pseudo*-discrete values in the search space, as illustrated in Fig. 1. To reduce the optimization time, the ODM performs local search of the best combination around the best ant and the decoupled technique [20] for optimization of the power conversion stage (PCS) and feedback network (FN) shown in Fig. 2 separately is applied. The eACO also takes the component tolerances into consideration in evaluating the fitness of each ant. The proposed algorithm has been successfully used to optimize the design of a buck regulator. The predicted results are compared with the published results

The work described in this paper was supported by the National Science Foundation of China Project No. 60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

available in the literature and are verified with experimental measurements.

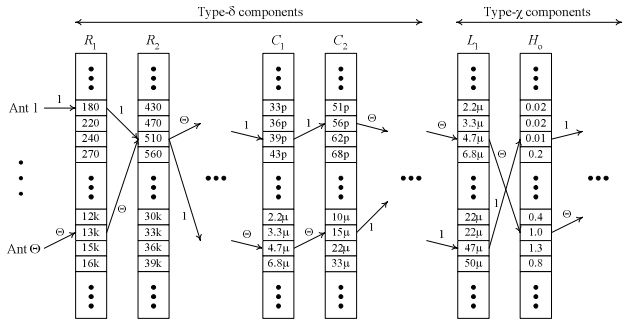


Fig. 1 Illustration of the ACO in finding various combinations of component values.

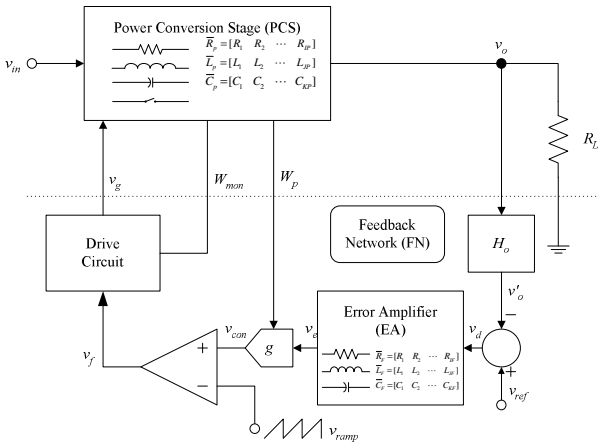


Fig. 2 Block diagram of power electronics circuits.

II. Formulation of the Data Structure and Fitness Function

As illustrated in Fig. 1, there are Θ ants to search different combinations of the components in the optimization algorithm. The data structure of each ant is described as below.

A. Data Structure

The circuit components are categorized into two types, type-δ components have discrete values in practice such as resistors and capacitors which are manufactured accurately to within a specified tolerance of their nominal value, type-χ components have continuous values such as inductors and voltage gains, which are manufactured accurately to within a specified accuracy. Each ant contains indexes referencing A_q to available values stored in the database for the chosen type-δ and type-χ components, and its fitness value. For the *q*-th ant,

$$A_q = [\delta^q \quad \chi^q \quad \Phi^q], \qquad q \in [1 \quad \Theta] \tag{1}$$

where δ^q is the index vector pointing to a database of the available values for the M type-δ components in the circuit, χ^q is the index vector pointing to a database of the available values for the N type-χ components in the circuit, and Φ^q is the fitness value of the q-th ant, δ^q and χ^q are defined as

$$\delta^q = [d^{q1} \quad d^{q2} \quad \cdots \quad d^{qi} \quad \ldots \quad d^{qM}] \tag{2}$$

$$\chi^q = [c^{q1} \quad c^{q2} \quad \cdots \quad c^{qj} \quad \ldots \quad c^{qN}] \tag{3}$$

where d^{qi} and c^{qj} are the indexes of the *i*-th type-δ component and *j*-th type-χ component, respectively, pointing to the chosen values available in the component database D. Thus,

$$D = \begin{bmatrix} \{d^1_{v1}, d^1_{p1}\}, & \cdots & \{d^1_{vm}, d^1_{pm}\}, & \cdots & \{d^1_{vS^{d^1}}, d^1_{pS^{d^1}}\}, & d^1_{tol} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \{d^i_{v1}, d^i_{p1}\}, & \cdots & \{d^i_{vm}, d^i_{pm}\}, & \cdots & \{d^i_{vS^{d^i}}, d^i_{pS^{d^i}}\}, & d^i_{tol} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \{d^M_{v1}, d^M_{p1}\}, & \cdots & \{d^M_{vm}, d^M_{pm}\}, & \cdots & \{d^M_{vm}, d^M_{pm}\}, & d^M_{tol} \end{bmatrix} \tag{4}$$

where $\{\bullet,\bullet\}$ is the parameter set of an available choice, S^{d^i} is the number of choices for the *i*-th type-δ component d^i_{vm} is the value of the *m*-th choice for d^{qi}, d^i_{pm} is the pheromone of the choice d^i_{vm}, and d^i_{tol} is the tolerance of the *i*-th component. The values of d^i_{v1}, d^i_{v2}, …, d^i_{vm}, …, and $d^i_{vS^{d^i}}$ are obtained from the manufacturers. Similarly, c^{qj} points to a parameter set stored in a component database C. The structure of C is shown as below,

$$C = \begin{bmatrix} \{c^1_{v1}, c^1_{p1}\}, & \cdots & \{c^1_{vn}, c^1_{pn}\}, & \cdots & \{c^1_{vS^{c^1}}, c^1_{pS^{c^1}}\}, & c^1_{tol} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \{c^j_{v1}, c^j_{p1}\}, & \cdots & \{c^j_{vn}, c^j_{pn}\}, & \cdots & \{c^j_{vS^{c^j}}, c^j_{pS^{c^j}}\}, & c^j_{tol} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \{c^N_{v1}, c^N_{p1}\}, & \cdots & \{c^N_{vn}, c^N_{pn}\}, & \cdots & \{c^N_{vS^{c^N}}, c^N_{pS^{c^N}}\}, & c^N_{tol} \end{bmatrix} \tag{5}$$

where S^{c^j} is the number of choices for c^{qj}, c^j_{vn} is the value of the *n*-th choice for c^{qj}, c^j_{pn} is the pheromone of the choice c^j_{vn}, and c^j_{tol} is the tolerance of the *j*-th component, c^j_{v1}, c^j_{v2}, …, c^j_{vk}, …, and $c^j_{vS^{c^j}} \in [C^j_{\min}, C^j_{\max}]$ are generated throughout the algorithm, $C^j_{\min}$ and $C^j_{\max}$ are the lower and upper limits of the search space for the *j*-th component.

B. Fitness Function for optimizing PCS

Each ant has a fitness value Φ^q showing the degree of attainment on the optimization objectives. The multi-objective functions used in [20] for optimizing PCS and FN are adopted. The fitness function for evaluating the ants for optimizing PCS is based on the following considerations,

1) the steady state error of the output voltage v_o within the required input voltage range $v_{in} \in [V_{in,\min}, V_{in,\max}]$ and output load range $R_L \in [R_{L,\min}, R_{L,\max}]$,
2) the operation constraints on circuit components, such as the maximum voltage and current stresses, ripple voltage and ripple current,
3) the steady state ripple voltage on v_o, and
4) the intrinsic factors associated with the components.

Each objective is mathematically described by an objective function. The formula of each objective function is given in [20]. Φ^q is the total sum of three components, including Φ^q_{-1}, Φ^q_0, and Φ^q_{+1}. Φ^q_{-1} is the fitness value of the Ant q when the component values are all reduced by the corresponding tolerance stored in the last columns in (4) and (5). Φ^q_0 is the fitness value of the Ant q when the components are in their corresponding nominal values. Φ^q_{+1} is the fitness value of Ant q when the component values are increased by the corresponding tolerance stored in the last columns of (4) and (5). Thus,

$$\Phi^q = (\Phi^q_{-1} + \Phi^q_0 + \Phi^q_{+1}) / 3 \qquad (6)$$

C. *Fitness function for optimizing FN*

The fitness function for optimizing FN is based on the following considerations:

1) same as the objective #1 for PCS,
2) the maximum overshoot and undershoot, and the settling time of v_o during the startup,
3) the steady state ripple voltage on v_o, and
4) the dynamic behaviors as in 2) during the input voltage and output load disturbances.

Again, the objective functions of the above objectives are defined in [20] and the tolerances of the components will be taken into calculations as in (6).

III. Optimization Procedures

With the help of the flowchart in Fig. 3, the optimization procedures consist of 8 steps.

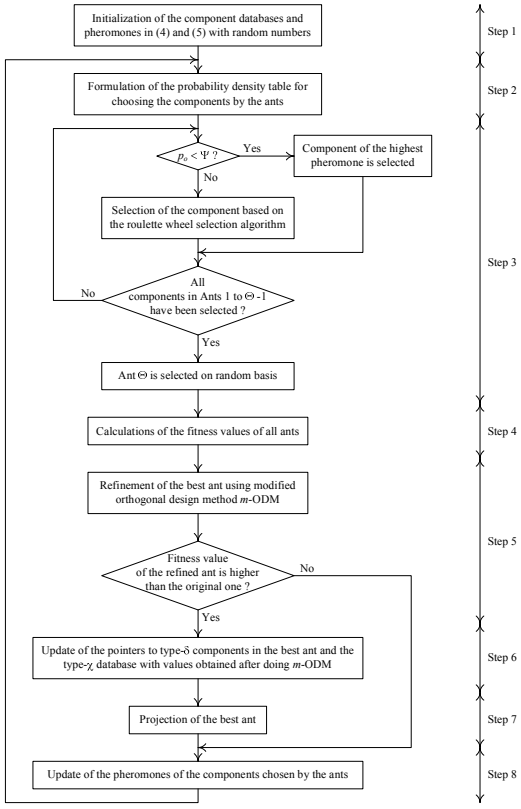


Fig. 3 Flowchart of the optimization procedures.

Step 1- Initialization

The values of d^i_{vm} (for i = 1, 2, …, M and m = 1, 2, …, S^{d^i}) in (4), and the tolerances d^i_{tol} in (4) and c^j_{tol} in (5) are initialized with the available values provided by the manufacturers and that of c^j_{vn} (for j = 1, 2, …, N and n = 1, 2, …, S^{c^j}) in (5) are initialized evenly over the search space and the number of choices assigned. Thus,

$$c^j_{vn} = (n-1)\,[(C^j_{\max} - C^j_{\min})] / S^{c^j} + C^j_{\min} \qquad (7)$$

The indexes d^{qi} in (2) and c^{qj} in (3) are randomly chosen. All pheromones d^i_{pm} in (4) are assigned with the same random number T_{d0}. All pheromones c^j_{pn} in (5) are assigned with the same random number T_{c0}. T_{d0} and T_{c0} will be used as parameters for updating the pheromones in each generation.

Step 2- Formation of the probability density table for choosing the components by the ants

A probability density table Π containing the probabilities of all components chosen in all ants that will be chosen again in the next generation is formulated. Π is in the form of

$$\Pi = \begin{bmatrix} \{d^1_1, p^1_{d1}, p^1_{cd1}\}, & \cdots & \{d^1_m, p^1_{dm}, p^1_{cdm}\}, & \cdots & \{d^1_{S_{d1}}, p^1_{dS_{d1}}, p^1_{cdS_{d1}}\} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \{d^i_1, p^i_{d1}, p^i_{cd1}\}, & \cdots & \{d^i_m, p^i_{dm}, p^i_{cdm}\}, & \cdots & \{d^i_{S_{di}}, p^i_{dS_{di}}, p^i_{cdS_{di}}\} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \{d^M_1, p^M_{d1}, p^M_{cd1}\}, & \cdots & \{d^M_m, p^M_{dm}, p^M_{cdm}\}, & \cdots & \{d^M_{S_{dM}}, p^M_{dS_{dM}}, p^M_{cdS_{dM}}\} \\ \{c^1_1, p^1_{c1}, p^1_{cc1}\}, & \cdots & \{c^1_n, p^1_{cn}, p^1_{ccn}\}, & \cdots & \{c^1_{S_{c1}}, p^1_{cS_{c1}}, p^1_{ccS_{c1}}\} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \{c^j_1, p^j_{c1}, p^j_{cc1}\}, & \cdots & \{c^j_n, p^j_{cn}, p^j_{ccn}\}, & \cdots & \{c^j_{S_{cj}}, p^j_{cS_{cj}}, p^j_{ccS_{cj}}\} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \{c^N_1, p^N_{c1}, p^N_{cc1}\}, & \cdots & \{c^N_n, p^N_{cn}, p^N_{ccn}\} & \cdots & \{d^N_{S_{cN}}, p^N_{cS_{cN}}, p^N_{ccS_{cN}}\} \end{bmatrix} \qquad (8)$$

where d^i_m is a pointer of the m-th choice for the i-th type-δ component to the database D chosen by one or several ant(s), p^i_{dm} is the probability of the component pointed by d^i_m, p^i_{cdm} is the cumulative probability of the component pointed by d^i_m, S_{di} is the total number of the i-th type-δ components chosen by all ants, c^j_n is a pointer of the n-th choice for the j-th type-χ component to the database C, p^j_{cn} is the probability of the component pointed by c^j_n, p^j_{ccn} is the cumulative probability of the component pointed by c^j_n, and S_{cj} is the total number of the j-th type-χ components chosen by all ants. Both S_{di} and S_{cj} are less than or equal to Θ. p^i_{dm} and p^i_{cdm} are calculated by using the pheromones stored in (4) and the formulas of

$$p^i_{dm} = d^i_{p[d^i_m]} / \sum_{k=1}^{S_{di}} d^i_{p[d^i_k]} \qquad (9)$$

and $$p^i_{cdm} = \sum_{k=1}^{m} p^i_{dk} \quad (10)$$

where $d^i_{p[d^i_k]}$ is the pheromone of the *i*-th component pointed by d^i_k and is stored in (4). Similarly, p^j_{cn} and p^j_{ccn} are calculated by using the pheromones stored in (5) and the following formulas of

$$p^j_{cn} = d^j_{p[c^j_n]} / \sum_{l=1}^{S_{cj}} d^j_{p[c^j_l]} \quad (11)$$

and $$p^j_{ccn} = \sum_{l=1}^{n} p^j_{cl} \quad (12)$$

where $d^j_{p[c^j_l]}$ is the pheromone of the *j*-th component pointed by c^j_l and is stored in (5).

Step 3 - Selection of the Components for all Ants

The values of the components in Ants 1 to (Θ - 1) are chosen among the values pointed by the pointers (i.e., d^i_m and c^j_n) in $\prod$. In selecting the value of each component in each ant, a random number p_o = rand(0,1) is generated and compared with an activation threshold Ψ. If $p_o < \Psi$, the component with the highest pheromone will be selected. Otherwise, the following roulette wheel selection algorithm will be performed. The *i*-th type-δ component for Ant *q* is selected by firstly generating a random number p_d and then making d^{qi} equal d^i_m in (8) using the criterion of $p^i_{cd(m-1)} < p_d < p^i_{cdm}$. Similarly, the *j*-th type-χ component for Ant *q* is selected by firstly generating a random number p_c and then making c^{qi} equal c^j_n in (8) using the criterion of $p^i_{cd(m-1)} < p_c < p^i_{cdm}$. Thus, for ant from A_1 to $A_{\Theta-1}$, the above procedures are repeated for *i* = 1…*M* and *j* = 1…*N*. ($d^{\Theta i}$ and $c^{\Theta j}$) for Ant Θ are randomly selected.

Step 4 – Calculations of the fitness values of all ants

After the component values for all ants have been assigned, the fitness values Φ^q [eq. (1)] of all ants are calculated by the fitness functions described in Sec. II.

Step 5 – Refinement of the best ant using modified orthogonal design method

The ant with the highest fitness value, namely the best ant A_b (where q = b), will be refined with the following modified orthogonal design method (ODM), which is extended from the standard ODM in [35]. Thus,

$$A_b = [\delta^b \quad \chi^b \quad \Phi^b] \quad (13)$$

where $\delta^b = [d^{b1} \quad d^{b2} \quad \cdots \quad d^{bi} \quad \cdots \quad d^{bM}]$, $\chi^b = [c^{b1} \quad c^{b2} \quad \cdots \quad c^{bj} \quad \cdots \quad c^{bN}]$, and $\Phi^b > \Phi^q$ $\forall q \neq b$. The behaviors around the solution set of A_b are observed. For the type-δ components, component values pointed by one-step regression and one-step progression, i.e., component values pointed by $(d^{bi} - 1)$ and $(d^{bi} + 1)$, $\forall i = 1, ..., M$, will be studied. For the type-χ components, component values of one radius regression and progression, i.e., component values of $(c^j_{v[c^{bj}]} - r_j)$ and $(c^j_{v[c^{bj}]} + r_j)$, $\forall j = 1, ..., N$, will be studied. $c^j_{v[c^{bj}]}$ is the value of the *i*-th component pointed by c^{bj}. In the *g*-th generation, the radius $r_j(g)$ is determined by

$$r_j(g) = r_j(g-1), \; g > 1 \quad (14)$$

where $r_j(1) = \dfrac{C^j_{\max} - C^j_{\min}}{2\,\Theta}$.

Thus, with each component having three possible values, there are totally 3^{M+N} combinations. It will require taking lengthy computations of the fitness values of all combinations to locate the combination having the highest fitness value around combination of the best ant. In order to shorten the computation time to locate the best combination, only *P* (where $P << 3^{M+N}$) representative sets of combinations and the method is based on extending the orthogonal design method in [35], as follows.

Firstly, an orthogonal array *L* for the (*M* + *N*) components with three possible values for each component is constructed. *L* is defined as

$$L = [\bar{l}_1 \quad \bar{l}_2 \quad \ldots \quad \bar{l}_b \quad \ldots \quad \bar{l}_{M+N}] \quad (15)$$

where $\bar{l}_b = [l_{1,b} \quad l_{2,b} \quad \ldots \quad l_{P,b}]^T$ and $l_{a,b} \in$ [-1, 0, 1] for *a* = 1,...,*P* representing a one-step regression ($l_{a,b}$ = -1), value unchanged ($l_{a,b}$ = 0), and a one-step progression ($l_{a,b}$ =1). Each column in *L* (i.e., $\bar{l}_b$) represents the variation of the *b*-th component and each row in *L* represents one of the nine combinations. Secondly, as illustrated in Table I, the fitness values Φ_1, …, Φ_9 of the nine different combinations are determined. Fig. 4 shows the test points marked with black dots around A_b. The numbers beside the black dots represent the combination shown in Table I.

Table I Orthogonal array with *M* = 2 and *N* = 1

Combination	Type-δ		Type-χ	Fitness value
	d^{b1}	d^{b2}	c^{b1}	
1	-1	-1	-1	Φ_1
2	-1	0	0	Φ_2
3	-1	1	1	Φ_3
4	0	-1	0	Φ_4
5	0	0	1	Φ_5
6	0	1	-1	Φ_6
7	1	-1	1	Φ_7
8	1	0	-1	Φ_8
9	1	1	0	Φ_9
K_{-1}	$\Phi_1+\Phi_2+\Phi_3$	$\Phi_1+\Phi_4+\Phi_7$	$\Phi_1+\Phi_6+\Phi_8$	
K_0	$\Phi_4+\Phi_5+\Phi_6$	$\Phi_2+\Phi_5+\Phi_8$	$\Phi_2+\Phi_4+\Phi_9$	
K_1	$\Phi_7+\Phi_8+\Phi_9$	$\Phi_3+\Phi_6+\Phi_9$	$\Phi_3+\Phi_5+\Phi_7$	

Thirdly, the sums of the fitness values corresponding to the variation of the respective component are calculated. They are labeled as K_{-1}, K_0, and K_1. For example, the

value of K_{-1} for the first type-δ component (i.e., d^{b1}) in Table I equals ($\Phi_1 + \Phi_2 + \Phi_3$). Fourthly, the best point is predicted by comparing the values of K_{-1}, K_0, and K_1 of each component. The one having the highest fitness value will give the direction of changing the considered component value. For example, if K_1 in the first type-δ component is the highest among the three, the predicted change of the component value is (d^{b1} - 1). The procedure will be performed for all ($M + N$) components. Finally, the fitness value of the projected best Ant A_b' will be calculated. If it is smaller than the fitness value of A_b, the program will go to Step 8.

Step 6 – Update of δ^b and type-χ component database

δ^b in (13) will be replaced with the pointers to type-δ components in Ant A_b'. The component values, which are stored in C, pointed by χ^b in (13) will be replaced with the ones obtained in Ant A_b'.

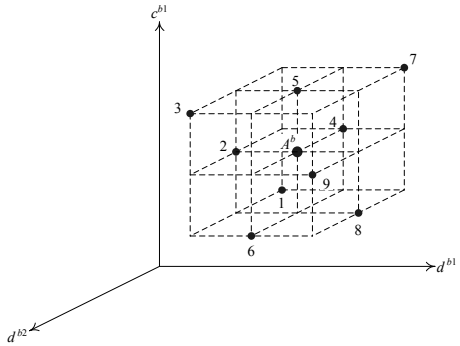


Fig. 4 Illustrative example showing the test points around A_b with the *m*-ODM.

Step 7 – Projection of a new Ant based on A_b and A_b'

A new Ant A_b'' is extrapolated in the same direction from A_b to A_b'. The type-δ components will be regressed, unchanged, or progressed one step. The radius of regression or progression of the type-χ components will be on random basis.

Then, the fitness value of Ant A_b'' will be calculated. If it is higher than that of Ant A_b, similar procedures as described in Step 6) will be performed.

Step 8 – Update of the pheromones of the components chosen by the ants

The pheromones of the components that are chosen by the ants are updated in two steps. The first step is to reduce the pheromones of all components that chosen by the ants. For the type-δ and type-χ components,

$$d^i_{p[d^{qi}]} = (1-\beta)\, d^i_{p[d^{qi}]} + \beta\, T_{\min} \quad , \quad \forall i=1,\ldots,M \;\; \& \;\; q=1,\ldots,\Theta \tag{16}$$

$$c^j_{p[c^{qj}]} = (1-\beta)\, c^i_{p[c^{qj}]} + \beta\, T_{\min} \, , \; \forall j=1,\ldots,N \;\&\; q=1,\ldots,\Theta \tag{17}$$

where $d^i_{p[d^{qi}]}$, which is stored in (4), is the pheromone of the *i*-th component pointed by the ant A^q, $[d^{qi}]$ is the value pointed by the pointer d^{qi} in (2), $c^j_{p[c^{qj}]}$, which is stored in (5), is the pheromone of the *j*-th component pointed by the ant A^q, $[c^{qj}]$ is the value pointed by the pointer c^{qj} in (3), β is the local update rate, and $T_{\min}$ is the lower bound of the pheromones. In order to increase the chance of being selected again in the next generation, the pheromones of the components that are chosen for A^b will be increased. For the type-δ and type-χ components,

$$d^i_{p[d^{bi}]} = (1-\rho)\, d^i_{p[d^{bi}]} + \rho T_{\max} \, , \;\; \forall i = 1, \ldots, M \tag{18}$$

$$c^j_{p[d^{bj}]} = (1-\rho)\, c^j_{p[d^{bj}]} + \rho T_{\max} \, , \;\; \forall j = 1, \ldots, N \tag{19}$$

where $d^i_{p[d^{bi}]}$, which is stored in (4), is the pheromone of the *i*-th component pointed by A^b, $[d^{bi}]$ is the value pointed by the pointer d^{bi} in (2)], $c^j_{p[c^{bj}]}$, which is stored in (5), is the pheromone of the *j*-th component pointed by A^b, $[c^{bj}]$ is the value pointed by the pointer c^{bj} in (2), $0<\rho<1$ is the global update rate, and $T_{\max}$ is the upper bound of the pheromone. It should be noted that the pheromones calculated in (16)-(19) will be corrected within [$T_{\min}$, $T_{\max}$], if their calculated values are outside the range.

IV. Design Example

The above method is applied to designing the buck regulator discussed in [20]. It consists of a buck converter and a proportional-plus-integral (PI) controller. The required specifications are as follows.

1) Input voltage range, v_{in}: 20V ~ 40V
2) Output load range, R_L: 5Ω - 10Ω
3) Nominal output voltage: 5V ± 1%
4) Switching frequency: 20kHz
5) Maximum settling time: 20ms

For the PCS, *L* and *C* are the design parameters. For the FN, all components are the design parameters. The optimized values of the components with GA [20] and proposed method are given in Table II. It can be observed that some of the values determined by GA are not available in a single component, but requires using several resistors or capacitors to fabricate the optimized values.

On the contrary, the values optimized by the proposed method are available practically. The optimization performances of the proposed method and the GA in [20] are compared. As the computation of the fitness value is the most time-consuming process, Fig. 5(a) shows the fitness value against the number of computations of the fitness value. The final fitness value obtained with the proposed method is 198 while the one with GA is 132. Thus, the proposed method can achieve a higher fitness value than the one with GA. Moreover, the proposed method can achieve a high fitness value with less number of computations. Fig. 5(b) shows the performance of the

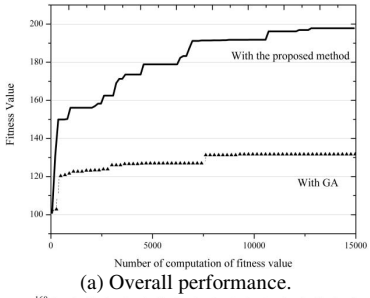
two methods in the first 1000 computations of the fitness value. The proposed method has already obtained a fitness value of 130 in 210 computations. The optimized component values with GA [19] and the proposed method are summarized in Table II. The component values shown in Table II have been applied to a practical circuit shown in Fig. 6, in which a controller TL494 is used. The experimental results of the converter with the GA-optimized and the proposed method have been compared. Fig. 7 shows the transient responses (including v_o and i_L) of the converter with the component values optimized by GA, when v_{in} is changed from 20V to 40V. The settling time is about 40ms in the testing conditions. Fig. 8 shows the transient responses of the converter when R_L is changed from 5Ω to 10Ω with v_{in} =20V. The settling times are 12ms under the testing condition. Figs. 9 and 10 show the waveforms corresponding to the testing conditions in Figs. 7 and 8, respectively, of the converter with the component values optimized by the proposed method. The settling times are 6ms in Fig. 9 and 5ms in Fig. 10, which are all shorter than the ones in the corresponding testing conditions. The results show that the component values optimized by the proposed method give better performance than that with GA. Moreover, the optimization scheme is general and is particularly suitable for designing PECs with complex structure. In addition, apart from the PI controllers as in the illustration, it is also applicable for optimizing complex controllers.

V. Conclusion

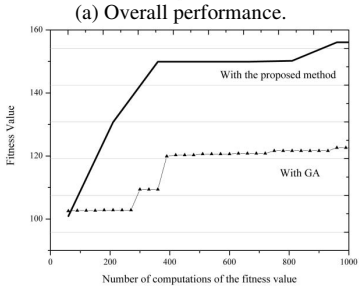
This paper presents an extended ant colony optimization algorithm for optimizing the component values in designing power electronic circuits, in order to satisfy the required performance indexes. It is unnecessary to conduct complicated mathematical analysis of the whole system and the values optimized are practically available. The proposed method has been illustrated with the design of a buck regulator. The simulation results have been compared with the ones of the converter using the component values optimized by genetic algorithms. The results show that the proposed method is faster and gives higher fitness value. Experimental results of a practical circuit have also demonstrated better performance, as compared with the converter using the component values optimized by GA.

Table II Optimized component values with GA [20] & proposed method

Components		GA in [20]	Proposed method
PCS	L	194μH	265μH
	C	1054μF	1000μF
FN	R_1	1.09356kΩ	82Ω
	R_2	766.56kΩ	82kΩ
	R_{C3}	3.448 kΩ	470Ω
	R_4	6.535kΩ	560Ω
	C_2	5.863 μF	0.22μF
	C_3	0.461 μF	3.3μF
	C_4	1.089 μF	2.2μF



(a) Overall performance.



(b) Performance in the first 1000 computations.

Fig. 5 Performance comparison between GA and the proposed method.

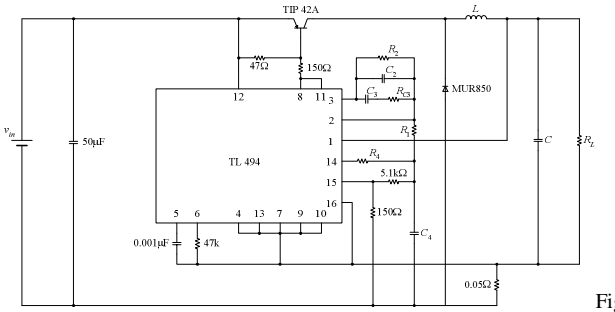


Fig. 6 Schematic diagram of the practical circuit.

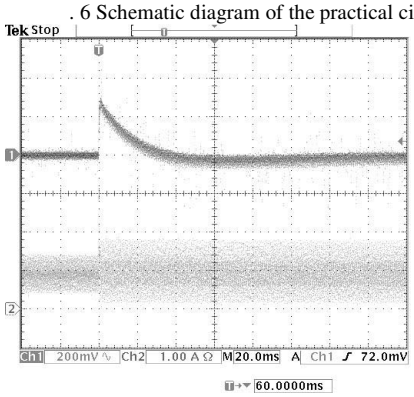


Fig. 7 v_{in} is changed from 20V to 40V. Experimental transient responses of v_o (200mV/div) and i_L (1A/div) with the values optimized by GA. (Timebase: 20ms/div).

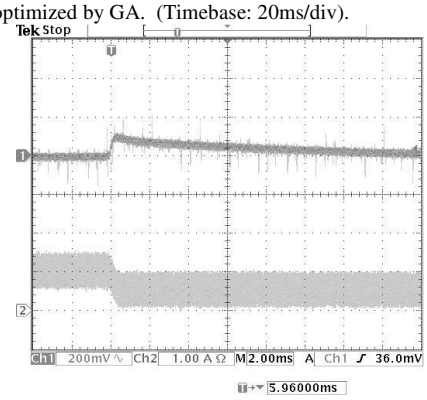


Fig. 8 R_L is changed from 5Ω into 10Ω. Experimental transient responses of v_o (200mV/div) and i_L (1A/div) with the values optimized by GA. (Timebase: 2ms/div).

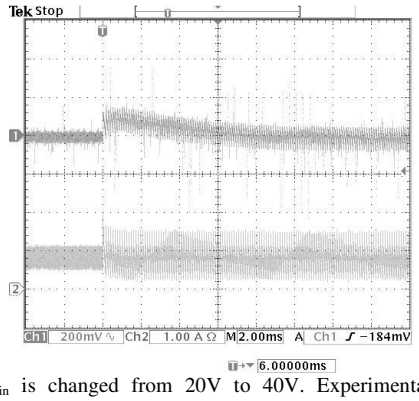


Fig. 9 v_{in} is changed from 20V to 40V. Experimental transient responses of v_o (200mV/div) and i_L (1A/div) with the values optimized by the proposed method. (Timebase: 2ms/div).

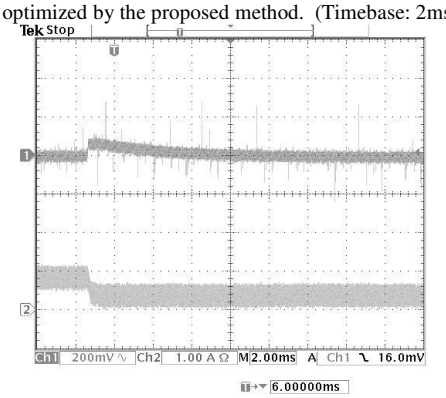


Fig. 10 R_L is changed from 5Ω into 10Ω. Experimental transient responses of v_o (200mV/div) and i_L (1A/div) with the values optimized by the proposed method. (Timebase: 2ms/div).

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.

[2] R. D. Middlebrook and S. Cuk, Advances in Switched-Mode Power Conversion, Pasadena, California, TESLACO, 1983.

[3] J. Kassakian, M. Schlecht, and G. Verghese, Principles of Power Electronics, Addison-Wesley, 1992.

[4] W. Ericksons and D. Maksimovic, Fundamentals of Power Electronics, New York: Chapman and Hall, May 1997.

[5] N. Mohan, T. M. Undeland, and W. P. Robbins, Power Electronics, Converters, Applications and Design, John Wiley & Sons, 2001.

[6] G. J. Sussman and R. M. Stallman, "Heuristic techniques in computer-aided circuit analysis," IEEE Trans. Circuits Systs., vol. 22, pp. 857-865, Nov. 1975.

[7] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," IEEE Trans. Computer-Aided Design, vol. 8, pp. 1247-1266, 1989.

[8] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX," IEEE Trans. Computer-Aided Design, vol. 15, pp. 273-294, Mar. 1996.

[9] L. P. Huelsman, "Optimization - a powerful tool for analysis and design," IEEE Trans. Circuits Systs. I, vol. 40, no. 7, pp. 431-439, Jul. 1993.

[10] R. E. Massara, Optimization Methods in Electronic Circuit Design, New York: Longman Scientific & Technical.

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

[12] M.D.L. del Casale, N. Femia, P. Lamberti, V. Mainardi, "Selection of optimal closed-loop controllers for DC-DC voltage regulators based on nominal and tolerance design," IEEE Trans. Ind. Electron., vol. 51, no. 4, pp. 840-849, Aug. 2004.

[13] K. S. Kostov and J. Kyyra, "Genetic algorithm optimization of peak current mode controlled buck converter," in Proc. IEEE mid-summer workshop on soft-computing in industrial applications, June 2005, pp. 111-116.

[14] K. S. Kostov and J. Kyyra, "Genetic algorithm controller optimization for SMPS," in Proc. IEEE Int. Conf. Systems, Man and Cybernetics, vol. 3, Oct. 2005, pp. 2182 – 2187.

[15] R. Wai and C. Tu, "Design of total sliding-mode-based genetic algorithm control for hybrid resonant-driven linear piezoelectric ceramic motor," IEEE Trans. Power Electron., vol. 22, no. 2, pp. 563-575, Mar. 2007.

[16] R. Wai and C. Tu, "Development of Lyapunov-based genetic algorithm control for linear piezoelectric ceramic motor drive," IEEE Trans. Ind. Electron., vol. 54, no. 5, pp. 2566-2582, Oct. 2007.

[17] P. Zanchetta, P. Wheeler, J. Clare, M. Bland, L. Empringham, and D. Katsis, "Control Design of a Three-Phase Matrix-Converter-Based AC–AC Mobile Utility Power Supply," IEEE Trans. Ind. Electron., vol. 55, no. 1, pp. 209-217, Jan. 2008.

[18] B. Ozpineci, L. Tolbert, and J. Chiasson, "Harmonic optimization of multilevel converters using genetic algorithms," IEEE Power Electron. Lett., vol. 3, no. 3, pp. 92-95, Sept. 2005.

[19] K. Shi and L. Hui, "Optimized PWM strategy based on genetic algorithms," IEEE Trans. Ind. Electron., vol. 52, no. 5, pp. 1458-1461, Oct 2005.

[20] 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.

[21] M. Liserre, A. Dell'Aquila, and F. Blaabjerg, "Genetic algorithm-based design of the active damping for an LCL-filter three-phase active rectifier," IEEE Trans. Power Electron., vol. 19, no. 1, pp. 76-86, Jan 2004.

[22] Y. Yang and X. Yu, "Cooperative Coevolutionary Genetic Algorithm for Digital IIR Filter Design," IEEE Trans. Ind. Electron., vol. 54, no. 3, pp. 1311-1318, Jun 2007.

[23] Y. Lee, W. Wang, and T. Kuo, "Soft Computing for Battery State-of-Charge (BSOC) Estimation in Battery String Systems," IEEE Trans. Ind. Electron., vol. 55, no. 1, pp. 229-239, Jan 2008.

[24] H. Surmann, "Genetic optimization of a fuzzy system for charging batteries," IEEE Trans. Ind. Electron., vol. 43, no. 5, pp. 541-548, Oct 1996.

[25] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29–41, Feb. 1996.

[26] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53–66, Apr. 1997.

[27] M. Dorigo and T. Stützle, Ant Colony Optimization. Cambridge, MA: MIT, Jul. 2004.

[28] M. Dorigo and G. D. Caro, "Ant colony optimization: A new metaheuristic," in Proc. IEEE Congr. Evol. Comput., Jul. 1999, vol. 2, pp. 1470–1477.

[29] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., vol. 6, no. 4, pp. 321–332, Aug. 2002.

[30] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load balancing: survey and new directions," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[31] C. Juang, C. Lu, C. Lo, and C. Wang, "Ant Colony Optimization Algorithm for Fuzzy Controller Design and Its FPGA Implementation," IEEE Trans. Ind. Electron., vol. 55, no. 3, pp. 1453-1462, Mar 2008.

[32] J. Gomez, P. De Oliveira, J. Yusta, R. Villasana, and A. Urdaneta, "Ant colony system algorithm for the planning of primary distribution circuits," IEEE Trans. Power Systems, vol. 19, no. 2, pp. 996-1004, May 2004.

[33] S. Favuzza, G. Graditi, M. Ippolito, and E. Sanseverino, "Optimal Electrical Distribution Systems Reinforcement Planning Using Gas Micro Turbines by Dynamic Ant Colony Search Algorithm," IEEE Trans. Power Systems, vol. 22, no. 2, pp. 580-587, May 2007.

[34] J. Vlachogiannis, N. Hatziargyriou, and K. Lee, "Ant Colony System-Based Algorithm for Constrained Load Flow Problem," IEEE Trans. Power Systems, vol. 20, no. 3, pp. 1241-1249, Aug. 2005.

[35] A. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays, Theory and Applications*, Springer-Verlag New York, Inc, 1999.