IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 11, NO. 3, JUNE 2007

Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms

Jun Zhang, *Member, IEEE*, Henry Shu-Hung Chung, *Senior Member, IEEE*, and Wai-Lun Lo, *Member, IEEE*

***Abstract*—Research into adjusting the probabilities of crossover and mutation p_m in genetic algorithms (GAs) is one of the most significant and promising areas in evolutionary computation. p_x and p_m greatly determine whether the algorithm will find a near-optimum solution or whether it will find a solution efficiently. Instead of using fixed values of p_x and p_m, this paper presents the use of fuzzy logic to adaptively adjust the values of p_x and p_m in GA. By applying the K-means algorithm, distribution of the population in the search space is clustered in each generation. A fuzzy system is used to adjust the values of p_x and p_m. It is based on considering the relative size of the cluster containing the best chromosome and the one containing the worst chromosome. The proposed method has been applied to optimize a buck regulator that requires satisfying several static and dynamic operational requirements. The optimized circuit component values, the regulator's performance, and the convergence rate in the training are favorably compared with the GA using fixed values of p_x and p_m. The effectiveness of the fuzzy-controlled crossover and mutation probabilities is also demonstrated by optimizing eight multidimensional mathematical functions.**

***Index Terms*—Evolutionary computation, fuzzy logics, genetic algorithms (GA), power electronics.**

I. Introduction

THE CONVENTIONAL approach to circuit optimization is to develop a formal model that can resemble actual circuit responses closely, but is solvable by means of available mathematical methods, such as linear and nonlinear programming. In the area of power electronics, state-space averaging and the variants [1]–[3] have been the dominant modeling techniques since 1970. By recognizing that power electronic circuits (PECs) typically have output filter cutoff frequency that is much lower than the switching frequency, linear time-invariant models, such as the control-to-output or input-to-output transfer functions, can be formulated to approximate the time-variant and piecewise-linear properties of the circuits. Although this approach has been proven to be very successful in many applications, it has the drawbacks of oversimplifying the circuit behaviors and of having limitations on particular operating mode and control schemes. As a circuit has been converted into a mathematical model and its state variables have been averaged, no detailed information about the exact waveforms and the response profiles can be obtained. Circuit designers would sometimes find it difficult to predict precisely the circuit responses under large-signal variations [3].

As power electronics technology continues to develop, a large number of combinatorial issues, including circuit complexity, static and dynamic responses, thermal problems, electromagnetic compatibility, control schemes, costing, etc., are associated. A plethora of such multimodal functions exist in a PEC. In particular, there is a growing need for automated synthesis that starts with high-level statements of the desired behaviors and optimizes the circuit component values for meeting required specifications.

Optimization strategies that are based on satisfying constrained equations might be subject to becoming trapped into local minima, leading to suboptimal parameter values, and thus, having a limitation on operating in large, multimodal, and noisy spaces. Since 1950, other strategies that employ Darwin's evolution theory have been proposed [4]–[6]. The most significant advantage of using this evolutionary search lies in the gain of flexibility and adaptability to the task at hand and the global search characteristics. Among various evolutionary computation methods (ECM), genetic algorithms (GA), which have been applied to many optimization problems [7], [8], employ a random, yet directed, search for locating the global optimal solution. They are superior to gradient descent techniques, as the search is not biased towards the local optimal solution. They differ from random sampling algorithms, as they can direct the search towards relatively prospective regions in the search space [9]. However, the usage of GA was progressed slowly in real applications. Apart from the shortcomings of early approaches, it was also largely due to the lack of powerful computer platforms at that time [10], [11].

Due to the recent advancements in computer technology, much research effort has been emphasized on developing new GA-based optimization methods. There are many new design schemes for analog circuits, like voltage reference circuit [12], transconductance amplifier [13], and analog circuit synthesis [14], [15]. Recently, GA have been applied to PEC optimization [16]–[18]. The circuit behaviors [16], [17] and controller functions [18] are described by well-defined mathematical functions with unknown optimal component values.

The parameters of the search space in GA are encoded in the form of a chromosome-like structure. A group of these chromosomes constitutes a population. An index of merit (fitness value) is assigned to each individual chromosome, according

Manuscript received June 15, 2005; revised January 11, 2006, April 4, 2006, and May 4, 2006.

J. Zhang is with the Department of Computer Science, Sun Yat-sen University, Guangzhou, China (e-mail: junzhang@ieee.org).

H. S.-H. Chung is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong (e-mail: eeshc@cityu.edu.hk).

W.-L. Lo is with the Department of Computer Science, Chu Hai College of Higher Education, Tsuen Wan, Hong Kong.

Digital Object Identifier 10.1109/TEVC.2006.880727

to a defined fitness function. A new generation is evolved by a selection technique, in which there is a larger probability of the fittest individuals being chosen. These chosen chromosomes are used as the parents in the construction of the next generation. A new generation is produced as a result of reproduction operators applied on parents. There are two main reproduction operators, namely, crossover and mutation. Crossover occurs only with some probability p_x. Notable crossover techniques include the single-point, the two-point, and the uniform types [19]. Mutation involves the modification of the value of each gene in the chromosome with some probability p_m. The role of mutation is to restore unexplored or lost genetic material into the population to prevent the premature convergence of the GA to suboptimal solutions. New generations are repeatedly produced until a predefined convergence level is reached.

The values of p_x and p_m significantly affect the behavior and the performance of the GA. A number of guidelines have been discussed in the literature for choosing them [20]–[22]. These generalized guidelines are inadequate because the optimal values of p_x and p_m are specific to the problem under consideration. Instead of using fixed p_x and p_m, some adaptive parameter control schemes that can relieve the burden of specifying the values of p_x and p_m have been proposed. In [22], a second-level GA is used to select p_x and p_m. Although this method can adjust p_x and p_m according to the solution distribution, it is computationally expensive. In [9], an adaptive GA is proposed. p_x and p_m are being varied depending on the fitness values of the solutions. Although its procedures of adjusting p_x and p_m are computationally efficient, the distribution of the chromosomes in the search space and searching maturity have not been considered.

This paper presents the use of fuzzy logic to adaptively adjust p_x and p_m. By applying the K-means algorithm [23], the distribution of the population in the search space is clustered in each generation. A fuzzy system is used to adjust the values of p_x and p_m. It is based on considering the relative size of the cluster containing the best chromosome and the one containing the worst chromosome. Both population distribution factor and computational efficiency are considered. The proposed adaptation method is applied to optimize a buck regulator that requires satisfying several static and dynamic requirements. The decoupled optimization technique, as described in [16], is used. The optimized component values, the regulator's performance, and the convergence rate are favorably compared with the GA using fixed values of p_x and p_m.

II. Brief Review on the GA Operation

The basic block diagram of a PEC includes the power conversion stage (PCS) and feedback network (FN) [16]. The PCS consists of I_P resistors (R), J_P inductors (L), and K_P capacitors (C). The FN consists of I_F resistors, J_F inductors, and K_F capacitors. The signal conditioner H_o converts the PCS output voltage v_o into a suitable form (i.e., v_o') for comparing with a reference voltage v_{ref}. Their difference v_d is then sent to an error amplifier (EA). The EA output v_e is combined with the feedback signals W_p, derived from the PCS parameters, such as the inductor current and input voltage, to give an output control voltage v_{con} after performing a mathematical function $g(v_e, W_p)$. v_{con} is then modulated by a pulse-width modulator to derive the required gate signals for driving the switches in the PCS. The values of all passive components in the PCS and the FN are contained in the following two vectors $\Theta_{\rm PCS}$ and $\Theta_{\rm FN}$:

$$\Theta_{\rm PCS} = [\overline{R}_P \quad \overline{L}_P \quad \overline{C}_P] \text{ and } \Theta_{\rm FN} = [\overline{R}_F \quad \overline{L}_F \quad \overline{C}_F] \quad (1)$$

where $\overline{R}_P = [R_1 \quad R_2 \quad \cdots \quad R_{I_P}]$, $\overline{L}_P = [L_1 \quad L_2 \quad \cdots \quad L_{J_P}]$, $\overline{C}_P = [C_1 \quad C_2 \quad \cdots \quad C_{K_P}]$, $\overline{R}_F = [R_1 \quad R_2 \quad \cdots \quad R_{I_F}]$, $\overline{L}_F = [L_1 \quad L_2 \quad \cdots \quad L_{J_F}]$, and $\overline{C}_F = [C_1 \quad C_2 \quad \cdots \quad C_{K_F}]$

$\Theta_{\rm PCS}$ and $\Theta_{\rm FN}$ are coded as vectors of floating point numbers of the same length as the solution vector. The precision of such an approach depends on the underlying machine. The format is generally better than that of the binary representation in conventional GA-training [24]. The same chromosome structure is defined in C-language for $\Theta_{\rm PCS}$ and $\Theta_{\rm FN}$ in the respective population. The search space of each component value is bounded within a predefined range.

Apart from satisfying the static and dynamic responses, the component values have to be optimized for other factors such as the physical size, cost, rating of the components, etc. In [16], $\Theta_{\rm PCS}$ and $\Theta_{\rm FN}$ are optimized separately by the GA. The PCS and FN are decoupled in the optimization. $\Theta_{\rm PCS}$ is optimized for the steady-state operating requirements of the PCS, including the input and output load range, steady-state error, and output ripple voltage. With the determined values of $\Theta_{\rm PCS}$, $\Theta_{\rm FN}$ is optimized for the whole-system steady-state and dynamic characteristics.

The procedures for optimizing PCS and FN are similar. Their major differences are the definitions of the fitness functions and population.

1) Step 1—Initialization: The population size N_p, the maximum number of generations $G_{\max}$, the probability of crossover operation p_x, the probability of mutation operation p_m, and the generation counter gen are initialized. All chromosomes CP_n are initialized with random numbers, which lie within the design limits. A population $U(0) = \{CP_n(0), n = 1, \ldots, N_p\}$ is created. The fitness values of all CP are calculated. The definitions of the fitness functions Φ can be found in [16]. The best chromosome in the initial generation $CP_B(0)$ having the highest fitness value (i.e., $\Phi[CP_B(0)] = \text{Max}\{\Phi[CP_n(0)], n = 1, \ldots N_p\}$), is selected as the reference for the next generation.

2) Step 2—Selection of Chromosomes: A selection process, which is based on applying the roulette wheel rule [24], is performed. It starts with the calculation of the fitness value $\Phi[CP_n(gen)]$, the relative fitness value $\Phi_r[CP_n(gen)]$, and the cumulative fitness value $\Phi_c[CP_n(gen)]$ for the $CP_n(gen)$

$$\Phi_r\left[CP_n(gen)\right] = \frac{\Phi\left[CP_n(gen)\right]}{\sum_{z=1}^{N_p} \Phi\left[CP_z(gen)\right]} \text{ and}$$

$$\Phi_c\left[CP_n(gen)\right] = \sum_{z=1}^{n} \Phi_r\left[CP_z(gen)\right]. \quad (2)$$

A random number $p \in [0,1]$ is generated and is compared with $\Phi_c[CP_n(gen)]$ for $n = 1 \ldots N_p$. If $\Phi_c[CP_{z-1}(gen)] < p \leq \Phi_c[CP_z(gen)]$, CP_z is selected to be a member of the new population. This selection process is repeated until N_p members have been selected for the new population. Chromosomes with higher fitness values will have higher probability to survive and might appear repeatedly in the new population [24].

3) Step 3—Reproduction Operations: A new chromosome will be reproduced by the crossover and mutation operations. For the crossover operation, two chromosomes are selected from the population. In order to determine whether a chromosome will undergo crossover, a random selection test (RST) is performed. The RST is based on generating a random number $p \in [0,1]$. If p is smaller than p_x, the chromosome will be selected. Another chromosome will then be chosen with the procedure. A crossover point is selected randomly with equal probability from one to the total number of components in the chromosomes. The genes after the crossover point will be exchanged to create two new chromosomes. The operations are repeated until all chromosomes have been considered.

The mutation operation also starts with a RST for each chromosome. If a generated random number $p \in [0,1]$ for a chromosome is smaller than p_m, the chromosome will undergo mutation. A random number will be generated for the chosen component with a value within the component limits. The procedures will be repeated until all chromosomes have been considered.

4) Step 4—Elitist Function: After calculating the fitness value of each chromosome, the best member $CP_B(gen)$ (the one having the highest fitness value), and the worst member $CP_w(gen)$ (having the lowest fitness values) will be identified. $CP_B(gen)$ will be compared with the best one in the last generation [i.e., $CP_B(gen-1)$]. If the fitness value of $CP_B(gen)$ is smaller than the one of $CP_B(gen-1)$, the content of $CP_B(gen-1)$ will substitute for $CP_B(gen)$. The content of $CP_B(gen-1)$ will substitute for $CP_w(gen)$. The GA cycle is then repeated from Step 2) again.

III. Adaptive Control of p_x and p_m

The values of p_x and p_m are fixed in typical GA, for example, $p_x = 0.85$ and $p_m = 0.25$ in [16] and are adjusted heuristically. However, biological evolution shows that p_x and p_m are dependent on the evolution state and should be adapted [25]. Thus, in order to enhance the training efficiency of [16], an adaptive approach to adjusting the values of p_x and p_m is proposed. The adjustment is based on considering the optimization state in the GA. Fig. 1 depicts the strategy for tuning p_x and p_m in four optimization states, including initial state, submaturing state, maturing state, and matured state [25]. In order to prevent premature convergence of the GA to a local optimum, it is essential to be able to identify whether the GA is converging to the optimum. The relative population distribution is used to define the optimization state in the proposed method. The first step is to partition the population into clusters. Those chromosomes having similar component vectors are grouped in the same cluster. The second step is to use a fuzzy system to fuzzify the relative sizes of the clusters. Adjustments of p_x and p_m are based on considering the relative size of the cluster containing the best chromosome and the one containing the worst chromosome. The procedures are described as follows.

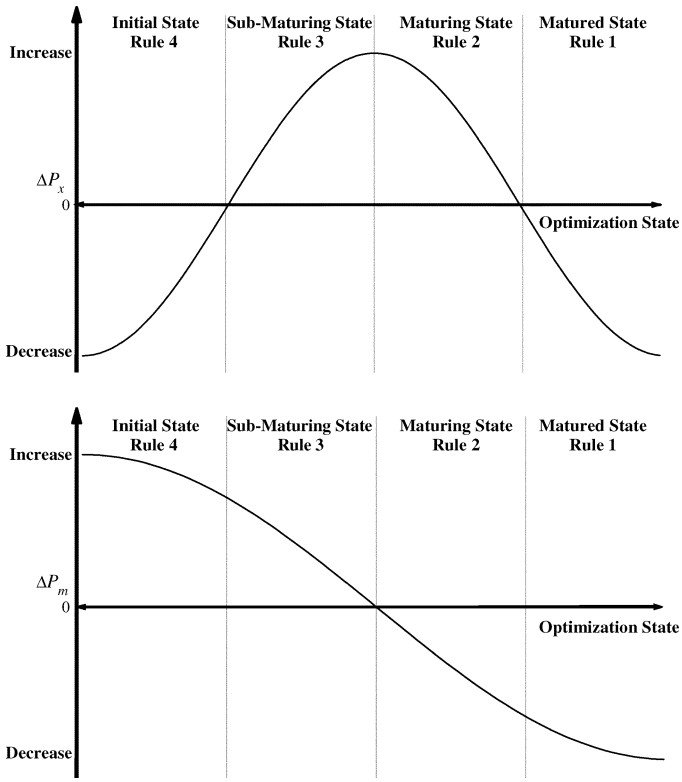


Fig. 1. Illustrations on adjusting p_x and p_m in different optimization phases.

A. Clustering of the Population

Although the K-means algorithm can only be used to partition suboptimal clusters [23], [26], it is sufficient for this particular application to depict the chromosome distribution. Assuming that the population is partitioned into K clusters. The clustering process is as follows.

Step 1) Choose K initial cluster centers $CP^1, CP^2, \ldots, CP^K$ randomly from the population $\{CP_1, CP_2, \ldots, CP_{Np}\}$.

Step 2) Assign CP_n, $n = 1, \ldots N_p$ to cluster C_j, $j \in \{1,2,\ldots,K\}$ if and only if

$$\|CP_n - CP^j\| < \|CP_n - CP^p\|, \quad p=1,2,\ldots,K, \text{ and } j \neq p \tag{3}$$

where $\|CP_n - CP^j\|$ is the distance between CP_n and CP^j.

Step 3) Compute new cluster centers $CP^{1,*}, CP^{2,*}, \ldots CP^{K,*}$ as follows:

$$CP^{j,*} = \frac{1}{M_j} \sum_{CP_n \in C_j} CP_n, \quad j = 1,2,\ldots,K \tag{4}$$

where M_j is the number of elements belonging to cluster C_j.

Step 4) If $CP^{j,*} = CP^j$, $j = 1,2,\ldots,K$, the process will be terminated. $CP^1, \ldots, CP^K$ are chosen as the cluster centers. Otherwise, assign each CP^j with $CP^{j,*}$, $j = 1,2,\ldots,K$, and step 2) will be started again.

The size of the cluster G_B (which contains the best chromosome) and the size of the cluster G_W (which contains the worst chromosome) are normalized by the difference between

TABLE I
STRATEGY OF TUNING THE VALUES OF p_x AND p_m

Size of the cluster containing the WORST chromosome	Size of cluster containing the BEST chromosome: Small	Size of cluster containing the BEST chromosome: Large
Large	p_x decrease, p_m increase	p_x Increase, p_m decrease
Small	p_x increase, p_m increase	p_x decrease, p_m decrease

the sizes of the largest cluster $G_{\max}$ and the smallest cluster $G_{\min}$. Mathematically

$$\hat{G}_B = \frac{G_B - G_{\min}}{G_{\max} - G_{\min}} \tag{5}$$

and

$$\hat{G}_W = \frac{G_W - G_{\min}}{G_{\max} - G_{\min}} \tag{6}$$

where $\hat{G}_B$ and $\hat{G}_W$ are the normalized values of G_B and G_W, respectively, ranging from zero to one. If the population is partitioned equally, (5) and (6) may be undefined because $G_{\max} = G_{\min}$. This condition can be avoided by explicitly checking its occurrence in the algorithm. Another way is to make the population size N_p be unequal to an integer multiple of K.

B. Rules for Tuning the Values of p_x and p_m

Tuning the values of p_x and p_m in the proposed fuzzy inference system is based on considering the relative cluster sizes of G_B and G_W (i.e., $\hat{G}_B$ and $\hat{G}_W$). The following three heuristic guidelines are used to formulate the fuzzy rules.

1) Is it necessary to enhance or suppress reproduction of chromosomes that are outside existing clustering distribution? This is related to the need of migrating searching direction from the existing cluster centers.
2) Is it necessary to enhance or suppress reproduction of chromosomes that are within existing clustering distribution? This is related to the need of refining solutions around the cluster centers.
3) Is it necessary to combine the guidelines 1) and 2) together?

Based on the above considerations, the following four rules for tuning p_x and p_m are defined and are tabulated in Table I.

Rule 1: The best chromosome is in the largest cluster, while the worst chromosome is in the smallest cluster. The values of p_x and p_m are reduced.

The training process is considered to be in the matured state. A large number of chromosomes with similar component vectors have swarmed together in the search space. The best member CP_B is possibly the solution for the optimization problem. The chance of reproducing new chromosomes through crossover and mutation across clusters is made smaller than the previous generation. The values of p_x and p_m will then be reduced. However, there are possibilities that CP_B is trapped into a local or suboptimal solution. Thus, it is necessary to check if the current best candidate is at a local optimal point. Rules 3) and 4) are designed to achieve this objective.

TABLE II
FUZZY CONTROL RULES FOR TUNING THE VALUES OF p_x AND p_m

Rule for δp_x
Rule 1=(Rule (1,0)): If ($\hat{G}_B$ is PB) and ($\hat{G}_W$ is PS) then δp_x=NB
Rule 2=(Rule (1,1)): If ($\hat{G}_B$ is PB) and ($\hat{G}_W$ is PB) then δp_x=PB
Rule 3=(Rule (0,0)): If ($\hat{G}_B$ is PS) and ($\hat{G}_W$ is PS) then δp_x=PB
Rule 4=(Rule (0,1)): If ($\hat{G}_B$ is PS) and ($\hat{G}_W$ is PB) then δp_x=NB

Rule for δp_m
Rule 1=(Rule (1,0)): If ($\hat{G}_B$ is PB) and ($\hat{G}_W$ is PS) then δp_m=NB
Rule 2=(Rule (1,1)): If ($\hat{G}_B$ is PB) and ($\hat{G}_W$ is PB) then δp_m=NB
Rule 3=(Rule (0,0)): If ($\hat{G}_B$ is PS) and ($\hat{G}_W$ is PS) then δp_m=PB
Rule 4=(Rule (0,1)): If ($\hat{G}_B$ is PS) and ($\hat{G}_W$ is PB) then δp_m=PB

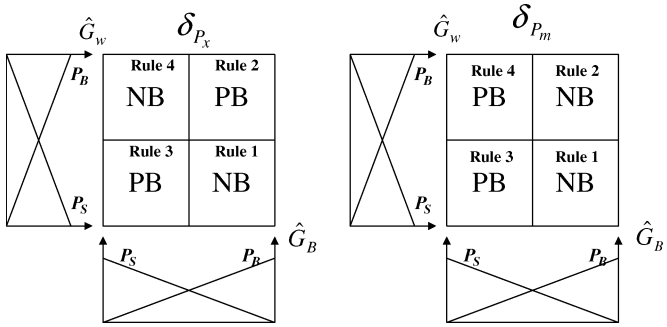


Fig. 2. Membership functions used in the fuzzy system.

If a newly generated chromosome has a higher fitness value than CP_B, $\hat{G}_B$ will be diminished and the value of p_m will be increased through Rule 3) and Rule 4). This can avoid the solution trapping into a local optimal point.

Rule 2: G_B equals G_W. Both of them are the largest among others. The value of p_x is increased and the value of p_m is reduced.

The training process is considered to be in the maturing state. The searching direction of the GA is still undetermined. The situation is twofold. First, the GA has to explore new searching directions. Second, the cluster containing CP_B has to be swarmed. A viable way is to enhance the search through crossover. As both G_B and G_W are dominant groups and have similar sizes, it is still not clear if the current best candidate is already at the global optimal point. It is expected that a new searching direction can be derived from G_B, rather than enhancing the growth of a particular cluster randomly. Thus, the value of p_x is increased. Increased crossover probability can make the new generation keep better behavior of parents and search for new direction. At the same time, the value of p_m is reduced, so that the probability of reproducing good candidates outside existing clusters will be reduced. The chromosomes in the cluster containing CP_W might have chances to reproduce new chromosomes in other clusters, including the one with CP_B.

TABLE III
BENCHMARKING THE SEARCHING SPEED WITH FIXED AND FUZZY-CONTROLLED p_x AND p_m

Function	No. of simulations that have found the solution after the sampled generations					
	With Fixed p_x and p_m			With Fuzzy-controlled p_x and p_m		
	300	500	1000	300	500	1000
F_1	910	984	1000	1000	1000	1000
F_2	80	157	376	173	341	584
F_3	134	270	560	398	629	855
F_4	33	54	133	69	114	193
F_5	59	111	250	158	260	389
F_6	747	922	997	978	1000	1000
F_7	101	220	491	354	577	842
F_8	65	126	267	175	280	406

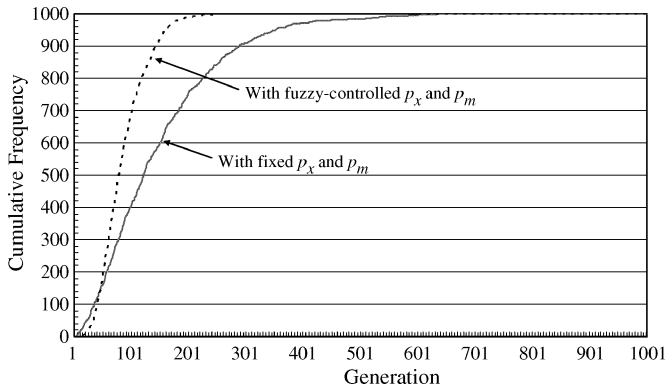


Fig. 3. Cumulative frequency of the solutions obtained in each generation with the respective setting for solving F_1.

Rule 3: G_B equals G_W. These are both the smallest in comparison to the others. The values of p_x and p_m are increased.

The training process is considered to be in the submaturing state. Similar to Rule 2), the searching direction is undetermined. However, the situation is that the population has not been swarmed to form a cluster with CP_B. Both G_B and G_W are minor groups in the current population. The overall searching process will be guided to explore new searching direction for enhancing the growth of the best candidates. In order to accelerate the generation of possible candidates within or outside the cluster containing CP_B, increments of the values of p_x and p_m are the viable way. Apart from reproducing chromosomes within the clusters, the generation of new chromosomes becomes possible.

Rule 4: The best chromosome is in the smallest cluster, while the worst chromosome is in the largest cluster. The value of p_x is reduced and the value of p_m is increased.

The training process is considered to be in the initial state. In order to reduce the chance of generating chromosomes with similar properties as CP_W, the value of p_x is reduced. At the same time, the chance of producing new candidates from the cluster containing CP_W has to be increased. Thus, the value of p_m is increased.

It is crucial to note that any decision to the above consideration should not lead to brute-stop or brute-force crossover and mutation operations. Otherwise, the philosophy of evolutionary computation will be lost.

C. Fuzzy-Based Tuning Mechanism for the Values of p_x and p_m

Inference of the values of p_x and p_m is based on a fuzzy-based tuning mechanism that consists of three major components, including fuzzification, decision-making, and defuzzification. $\hat{G}_B$ and $\hat{G}_W$ are the inputs to this inference system.

1) Fuzzification: Fuzzification is to map the input variables $\hat{G}_B$ and $\hat{G}_W$ into suitable linguistic values. As shown in (5) and (6), $\hat{G}_B$ and $\hat{G}_W$ are always positive. Two fuzzy subsets including positive small (PS) and positive big (PB) are defined. Each input variable is assigned to two membership values $\mu_{\rm PS}$ and $\mu_{\rm PB}$ corresponding to PS and PB fuzzy subsets. Fig. 2 illustrates the membership functions, which are linear in nature. For PS fuzzy subset

$$\mu_0(\hat{G}) = \mu_{\rm PS}(\hat{G}) = 1 - \hat{G} \qquad 0 \le \hat{G} \le 1 \tag{7}$$

where $\hat{G}$ equals $\hat{G}_B$ or $\hat{G}_W$, respectively.

For PB fuzzy subset

$$\mu_1(\hat{G}) = \mu_{\rm PB}(\hat{G}) = \hat{G} \qquad 0 \le \hat{G} \le 1. \tag{8}$$

In general, the number of fuzzy subsets depends on the required input resolution [27]. In this application, two fuzzy subsets are sufficient.

2) Decision-Making: Decision-making infers fuzzy control action from knowledge of the fuzzy rules and the linguistic variable definition. Table II shows the control rule table used, in which each entry corresponds to a control rule in Section III-B. The fuzzy inference method is illustrated in Fig. 2. As every values of $\hat{G}_B$ and $\hat{G}_W$ belong to two fuzzy subsets, four rules

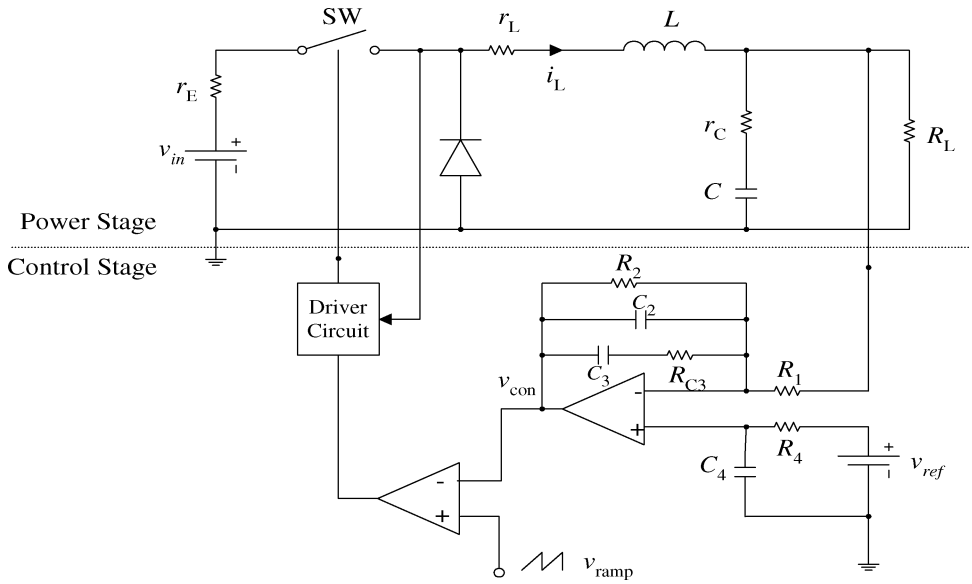


Fig. 4. Schematics of the buck regulator in [16].

including $(\mathrm{PS},\mathrm{PS})$, $(\mathrm{PS},\mathrm{PB})$, $(\mathrm{PB},\mathrm{PS})$, and $(\mathrm{PB},\mathrm{PB})$ have to be considered in each generation. Considering the rule $(\mathrm{PS},\mathrm{PB})$, a value $m_{0,1}$ is determined by algebraic multiplication fuzzy implication of μ_{PS} and μ_{PB}, where

$$m_{0,1} = \mu_{\mathrm{PS}}(\hat{G}_B)\mu_{\mathrm{PB}}(\hat{G}_W). \tag{9}$$

The same operation is applied for other rules. The union of all the fuzzy sets will be used to derive the changes of the values of p_x and p_m after defuzzification.

3) Defuzzification: Defuzzification is the process to convert the inferred fuzzy action to a crisp value. The output of the inference system is the changes of the values of p_x and p_m. The actual value is determined by adding $p_x(gen-1)$ and $p_m(gen-1)$ to the calculated change. That is

$$p_x(gen) = p_x(gen-1) + K_x\delta p_x(gen) \tag{10}$$

and

$$p_m(gen) = p_m(gen-1) + K_m\delta p_m(gen) \tag{11}$$

where K_x and K_m are chosen to keep the changes of the values of p_x and p_m within a tolerance percentage of the nominal level in each generation. Crisp values for δp_x and δp_m are calculated by applying the "*center of sum method*." The defuzzified output is calculated by the formulas of

$$\delta p_x = \frac{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1}\mu_i(\hat{G}_B)\mu_j(\hat{G}_W)y_{ij}}{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1}\mu_i(\hat{G}_B)\mu_j(\hat{G}_W)} \tag{12}$$

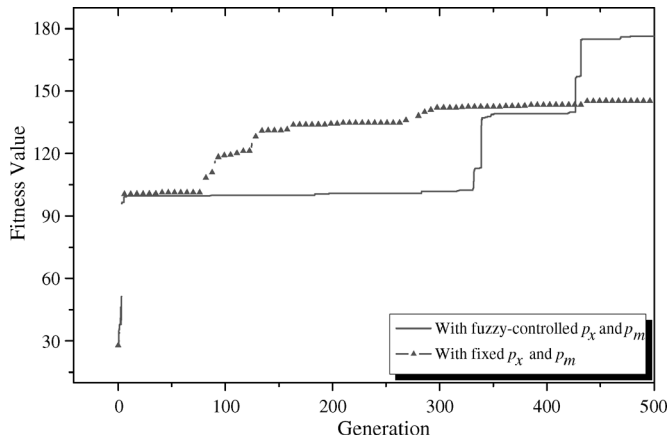


Fig. 5. Comparisons of the fitness values against the training generation using fixed and fuzzy-controlled p_x and p_m.

and

$$\delta p_m = \frac{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1}\mu_i(\hat{G}_B)\mu_j(\hat{G}_W)z_{ij}}{\sum_{i=0}^{i=1}\sum_{j=0}^{j=1}\mu_i(\hat{G}_B)\mu_j(\hat{G}_W)} \tag{13}$$

where y_{ij} is the center of the output fuzzy set of δp_x for Rule (i,j) and z_{ij} is the center of the output fuzzy set of δp_m for Rule (i,j). In this paper, the output fuzzy set is chosen to be singleton. That is, y_{ij} and z_{ij} will be either $+1$ or -1. They are governed by the rules in Table I. For example, $\delta p_x =$ 'PB' in Rule (0, 1), and thus $y_{ij} = +1$ is taken. Equations (10) and (11) are used in the GA. However, it should be noted that the values of p_x and p_m have limits. For example, as discussed in [28], $p_x \in [0.75, 0.95]$ and $p_m \in [0.005, 0.01]$.

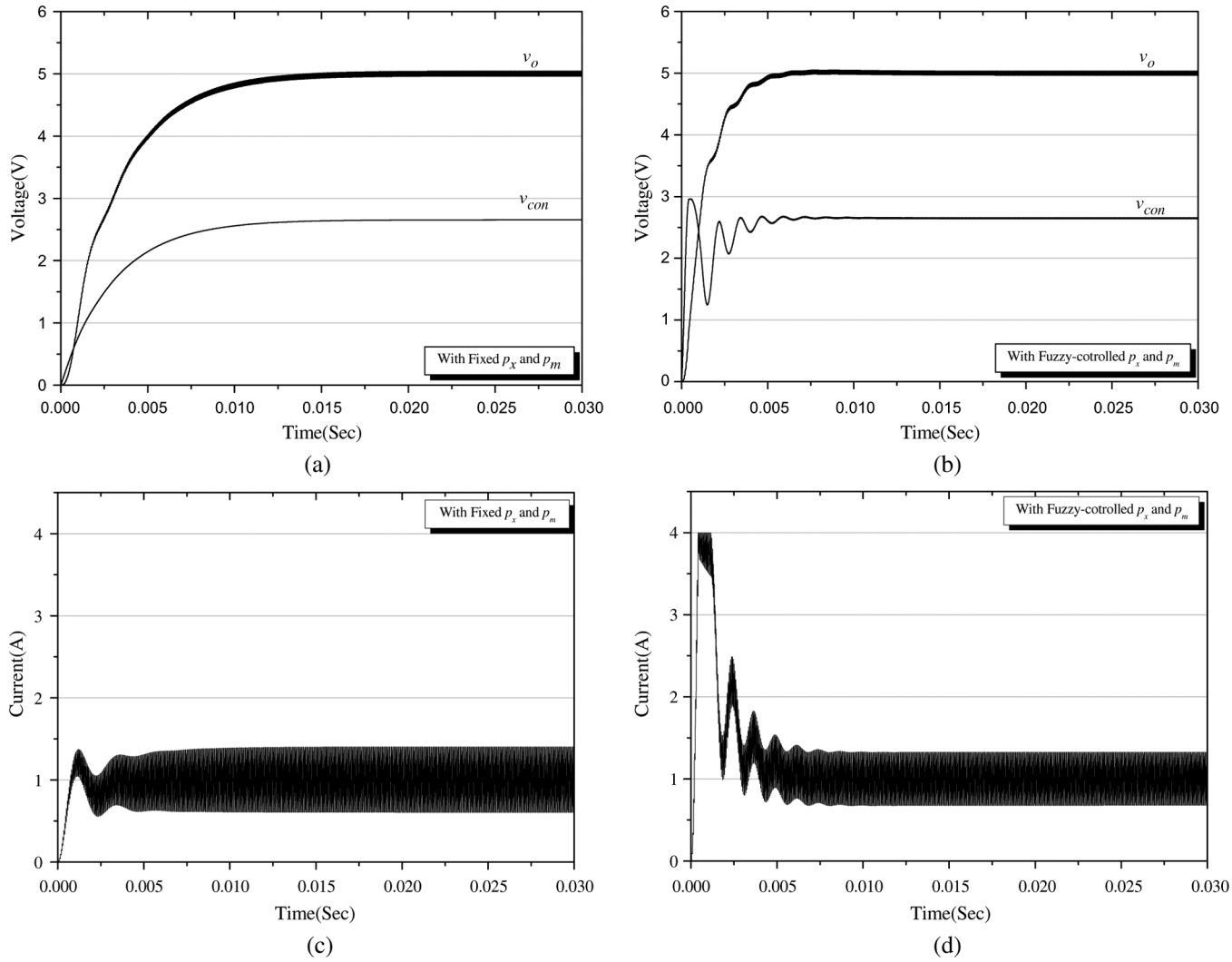


Fig. 6. Simulated startup transients when v_{in} is 20 V and R_L is 5 Ω. (a) v_o and v_{con} (with fixed p_x and p_m). (b) v_o and v_{con} (with fuzzy-controlled p_x and p_m). (c) i_L (with fixed p_x and p_m). (d) i_L (with fuzzy-controlled p_x and p_m).

IV. Examples and Comparisons

Two categories of examples have been studied. The first category is to optimize eight sets of mathematical functions, while the second one is to optimize the circuit parameters of a buck regulator.

A. Example 1—Mathematical Functions

Eight mathematical functions $F_1 \sim F_8$ are taken and are listed as follows:

$$F_1(x_1,x_2,x_3)=\sum_{i=1}^{3}x_i^2, \qquad x_i\in[-5.12,5.12] \tag{14}$$

$$F_2(x_1,x_2)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}, \quad x_i\in[-65.536,65.536] \tag{15}$$

where (see equation at the bottom of the page)

$$F_3(x)=\sum_{i=1}^{2}|x_i|+\prod_{i=1}^{2}|x_i|, \quad x_i\in[-100,100] \tag{16}$$

$$F_4(x_1,x_2)=\sum_{i=1}^{2}\left(\sum_{j=1}^{i}x_j\right)^2, \quad x_i\in[-100,100] \tag{17}$$

$$a_{ij}=\begin{bmatrix}-32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32\end{bmatrix}$$

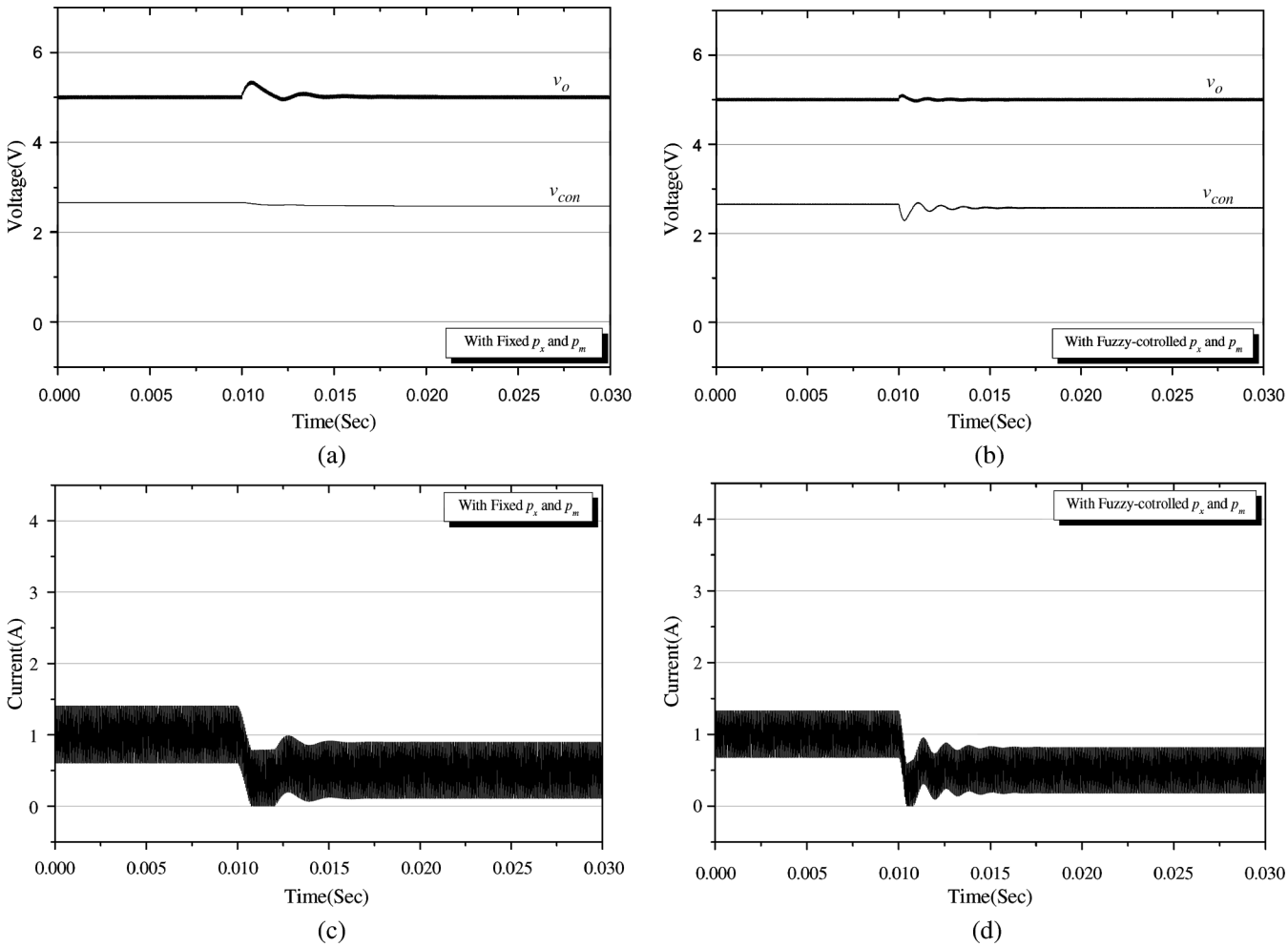


Fig. 7. Simulated transient responses when R_L is changed from 5 Ω into 10 Ω and v_{in} is 20 V. (a) v_o and v_{con} (with fixed p_x and p_m). (b) v_o and v_{con} (with fuzzy-controlled p_x and p_m). (c) i_L (with fixed p_x and p_m). (d) i_L (with fuzzy-controlled p_x and p_m).

$$F_5(x) = \max_i \left(|x_i|, 1 \le i \le 2\right), \quad x_i \in [-100, 100] \tag{18}$$

$$F_6(x) = \sum_{i=1}^{2} (x_i + 0.5)^2, \quad x_i \in [-100, 100] \tag{19}$$

$$F_7(x) = \sum_{i=1}^{2} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \quad x_i \in [-100, 100] \tag{20}$$

$$F_8(x) = \frac{1}{4000}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2} \cos\frac{x_i}{i} + 1, \quad x_i \in [-100, 100]. \tag{21}$$

The major goal of this study is to determine the values of x_i in all functions, so that the values of those functions are minimal within the search space of x_i. Each function is optimized by two settings of p_x and p_m. The first setting is that both p_x and p_m are fixed. The second setting is that both p_x and p_m are fuzzy-controlled. In each setting, one thousand simulations have been carried out to solve each function. Each simulation is initialized with different initial conditions. For the sake of comparison, the 1000 sets of initial conditions are the same for both settings in solving each function. Both settings have the same initial p_x and p_m, where $p_x = 0.6$ and $p_m = 0.01$, in solving the functions. The chromosome population size is 120. Fig. 3 compares the cumulative frequency of solutions obtained in each generation with the respective setting. 50%, 90%, and 100% of chances can be found in the 122nd, 290th, and 912th generations, respectively, with fixed p_x and p_m. With fuzzy-controlled p_x and p_m, 50%, 90%, and 100% of chances have been found in the 78th, 144th, and 264th generations, respectively. Solutions can be found at a lower generation with the proposed fuzzy-controlled p_x and p_m.

Table III benchmarks the searching speed of solving all functions shown in (14)–(21) with fixed and fuzzy-controlled p_x and p_m. The table shows the number of simulations that can determine the solution after 300, 500, and 1000 generations. For example, in solving F_1, 910 out of the 1000 simulations have found the solution after 300 generations and 984 simulations have found the solution after 500 generations with fixed p_x and p_m. On the other hand, all 1000 simulations have found the solution after 300 generations with the proposed fuzzy-controlled p_x and p_m.

It can be observed from Table III that GA with fuzzy-controlled p_x and p_m can generally search the solutions faster. These show the advantages of the proposed method.

B. Example 2—Design of a Buck Regulator

The proposed method is illustrated with the same example in [16]. The circuit schematic is shown in Fig. 4. The PCS is a classical buck converter and the FN is a proportional-plus-integral controller. In [16], $p_x(= 0.85)$ and $p_m(= 0.25)$ are fixed in the GAs. Fig. 5 shows the comparisons of the fitness values against the training generations with the fixed and the proposed fuzzy-controlled p_x and p_m. It can be seen that the fuzzy-controlled scheme can significantly improve the fitness values.

Fig. 6 shows the startup transients when the input voltage is 20 V and the output load is 5 Ω. The settling time with the proposed method is less than 10 ms, which is shorter than the design in [16]. Fig. 7 shows the transients when the output load is changed from 5 to 10 Ω. From the above results, it can be shown that the optimized circuit parameters with the proposed technique give better performances than the ones obtained in [16], showing the advantage of the proposed method.

V. Conclusion

A fuzzy-controlled crossover and mutation probabilities in GA for optimization of PECs has been proposed. They are determined adaptively for each solution of the population. It is in the manner that the probabilities are adapted to the population distribution of the solutions. This not only improves the convergence rate of the GA, but also prevents the solution from trapping into a local optimum point. A set of several mathematical functions and a buck regulator have been optimized. The results are favorably compared with the ones using GA with fixed probabilities of crossover and mutation.

Acknowledgment

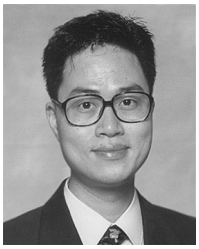
The authors would like to thank the associate editor and reviewers for their valuable comments and suggestions.

References

[1] K. K. Sum, *Switch Mode Power Conversion: Basic Theory and Design*. New York: Marcel Dekker, 1984.
[2] J. G. Kassakian, M. F. Schlecht, and G. C. Verghese, *Principles of Power Electronics*. Reading, MA: Addison-Wesley, 1991.
[3] Y. S. Lee, *Computer-Aided-Analysis of Switch-Mode Power Supplies*. New York: Marcel-Dekker, 1993.
[4] G. E. P. Box, "Evolutionary operation: A method for increasing industrial productivity," *J. Roy. Statist. Soc., C*, vol. 6, no. 2, pp. 81–101, 1957.
[5] H. J. Bremermann, "Optimization through evolution and recombination," in *Self-Organizing Systems*, M. Yovits, G. T. Jacobi, and G. D. Goldstein, Eds. Washington, DC: Spartan, 1962, pp. 93–106.
[6] C. Dimopoulos and A. M. S. Zalzala, "Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons," *IEEE Trans. Evol. Comput.*, vol. 4, no. 2, pp. 93–113, Jul. 2000.
[7] K. De Jong, "Are genetic algorithms function optimizers?," in *Parallel Problem Solving from Nature 2*. Amsterdam, The Netherlands: Elsevier, 1992, pp. 3–13.
[8] ——, "Genetics are NOT function optimizers," in *Foundations of Genetic Algorithms 2*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 5–17.
[9] M. Srinivas and L. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.
[10] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.
[11] T. Back, U. Hammel, and H. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 15–28, Apr. 1997.
[12] D. Nam, Y. Seo, L. Park, C. Park, and B. Kim, "Parameter optimization of a reference circuit using EP," in *Proc. 1998 IEEE Int. Conf. Evol. Comput.*, 1998, pp. 301–305.
[13] M. Wojcikowski, J. Glinianowicz, and M. Bialko, "System for optimisation of electronic circuits using genetic algorithm," in *Proc. IEEE Int. Conf. Electron., Circuits, Syst.*, 1996, vol. 1, pp. 247–250.
[14] N. N. Dhanwada, A. Nunez-Aldana, and R. Vemuri, "A genetic approach to simultaneous parameter space exploration and constraint transformation in analog synthesis," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1999, vol. 6, pp. 362–265.
[15] J. D. Lohn, S. P. Colombano, G. L. Haith, and D. Stassinopoulos, "A parallel genetic algorithm for automated electronic circuit design," in *Proc. Computational Aerosciences Workshop*, Feb. 2000, NASA Ames Research Center.
[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, H. Chung, S. Y. R. Hui, W. L. Lo, and A. Wu, "Decoupled optimization of power electronics circuits using genetic algorithm," in *Practical Handbook of Genetic Algorithms—Applications*. Boca Raton, FL: CRC Press, 2000, pp. 135–166.
[18] H. Chung, E. Tam, W. L. Lo, and S. Y. R. Hui, "An optimized fuzzy logic controller for active power factor corrector using genetic algorithms," in *Practical Handbook of Genetic Algorithms—Applications*. Boca Raton, FL: CRC Press, 2000, pp. 363–390.
[19] W. M. Spears and K. A. De Jong, "An analysis of multipoint crossover," in *Proc. Workshop of the Foundations of Genetic Algorithms*, 1990, pp. 301–315.
[20] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975.
[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison Wesley, 1989.
[22] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 16, no. 1, pp. 122–128, Jan. 1986.
[23] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison-Wesley, 1974.
[24] Z. Michalewicz, *Genetic Algorithms+Data Structure=Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.
[25] J. Reed, "Simulation of biological evolution and machine learning," *J. Theoret. Biol.*, vol. 17, pp. 319–342, 1967.
[26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
[27] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[28] J. Schaffer, R. Caruana, L. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 51–60.

Jun Zhang (M'02) received the Ph.D. degree in electrical engineering from City University of Hong Kong, Kowloon, in 2002.

From 2003 to 2004, he was a Brain Korean 21 Postdoctoral Fellow in the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). Since 2004, he has been with the SUN Yat-sen University, Guangzhou, China, where he is currently an Associate Professor with the Department of Computer Science. He has authored two research book chapters and over 30 technical papers in his research areas. His research interests include genetic algorithms, ant colony system, fuzzy logic, neural network, and nonlinear time series analysis and prediction.



Henry Shu-Hung Chung (M'95–SM'03) received the B.Eng. degree in electrical engineering and the Ph.D. degree from the Hong Kong Polytechnic University, Kowloon, in 1991 and 1994, respectively.

Since 1995, he has been with the City University of Hong Kong (CityU), Kowloon. He is currently a Professor with the Department of Electronic Engineering and Chief Technical Officer of e.Energy Technology Limited—an associated company of CityU. His research interests include time- and frequency-domain analysis of power electronic circuits, switched-capacitor-based converters, random-switching techniques, control methods, digital audio amplifiers, soft-switching converters, and electronic ballast design. He has authored four research book chapters, and over 200 technical papers including 90 refereed journal papers in his research areas, and holds ten patents.

Dr. Chung was awarded the Grand Applied Research Excellence Award in 2001 from the City University of Hong Kong. He was IEEE Student Branch Counselor and Track Chair of the Technical Committee on power electronics circuits and power systems of the IEEE Circuits and Systems Society from 1997 to 1998. He was Associate Editor and Guest Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART I: FUNDAMENTAL THEORY AND APPLICATIONS from 1999 to 2003. He is currently Associate Editor of the IEEE TRANSACTIONS ON POWER ELECTRONICS.

Wai-Lun Lo (M'02) received the B.Eng. degree in electrical engineering and the Ph.D. degree from the Hong Kong Polytechnic University, Kowloon, in 1991 and 1996, respectively.

He has been a Research Assistant and then a Research Associate in the Department of Electrical Engineering, Hong Kong Polytechnic University from 1996 to 1997. From 1997 to 1999, he was a Postdoctoral Fellow in the Department of Electrical Engineering, Hong Kong Polytechnic University. In 1999, he was with the Department of Electronic Engineering, City University of Hong Kong as a Research Fellow before joining the Department of Computer Science, Chu Hai College of Higher Education in September 1999. He is currently the Head of the Department of Computer Science, Chu Hai College of Higher Education. His research interest includes adaptive control, fuzzy control, intelligent control of complex nonlinear systems via neural network and applications of genetic algorithm.