Pseudo-Coevolutionary Genetic Algorithms for Power Electronic Circuits Optimization

Jun Zhang*, Henry S.H. Chung†, W.L. Lo‡, Eugene P.W. Tam†, J. J. Lee*, Angus K.M. Wu†,
*Dept of EECS, KAIST, Korea; †Dept of E.E., City University of Hong Kong; ‡ Chu Hai College,Hong Kong
junzhang@ieee.org

Abstract – This paper presents pseudo-coevolutionary genetic algorithms (GA's) for power electronic circuit (PEC) optimization. Circuit parameters are optimized through two parallel co-adapted GA-based optimization processes for power conversion stage and feedback network, respectively. Each process has tunable and untunable parametric vectors. The best candidate of the tunable vector in one process is migrated into the other process as untunable vector through a migration controller, in which the migration strategy is adaptively controlled by a first-order projection of the maximum and minimum bounds of the fitness value in each generation. Implementation of this method is suitable for systems with parallel computation capacity, resulting in considerable improvement of the training speed. Optimization of a buck regulator for meeting requirements under large-signal changes and at steady state is illustrated. Simulation predictions are verified with experimental results.

1. Introduction

As modern power electronics technology continues to develop, there is a growing need for automated synthesis that starts with high-level statements of desired behaviors and optimizes the component values for satisfying electrical specifications. Evolutionary computations such as genetic algorithms (GA's) which have emerged as a practical, robust optimization and search method [1]-[5], have been applied to optimize analog electronic circuits [2]-[4]. Recently, the method has been extended to optimize PEC's in [11], in which a decoupled optimization technique for design of switching regulator was proposed. The optimization process entails selection of the component values in the regulator to meet some static and dynamic requirements. A regulator is decoupled into a power conversion stage (PCS) and a feedback network (FN). The circuit component values in the PCS are optimized with the required static characteristics, whilst the ones in the FN are optimized with the required static and dynamic behaviors of the whole system. However, interactions between the PCS and FN in the optimization are relatively low with this training scheme.

As well as the classical serial GA's, extensive research has been conducted on the development of parallel GA's (PGA's) [7]-[9]. Coordinated GA's running in parallel result in a very powerful method for solving multi-modal function and high-dimension optimization problems. In particular multiple-population PGAs are the most sophisticated ones. They consist of several populations, in which individuals migrate occasionally. Parameters such as migration rates, communication topology, and population size affect their behaviors significantly. With enhancements in the multiple-population PGAs, cooperative evolutionary computations have been proposed in [8][9]. They provide rational opportunities for solutions to evolve in the form of interacting coadapted subpopulation (species). Apart from the interaction architecture, migration strategy is an influential factor affecting the performance.

Synchronous migration is the most dominant one, in which migration occurs at constant time or generation intervals. Another one is the asynchronous migration scheme, which allows individuals to migrate only after some events occur. Low migration rate is generally suitable for separable searching functions. Relatively isolated species are likely to converge to different solutions. Migration and recombination may combine partial solutions, resulting in low fitness as compared to classical GAs. If the migration is too fast, high potential individuals in the migrants may be too less to influence the overall search.

An asynchronous migration scheme using pseudo-coevolutionary GA's for design of PEC's is presented in this paper. By using the decoupled optimization technique in [6], component values of the PCS and FN are optimized by two coadapted evolutionary training processes. They are treated as two species and are evolved in parallel. Each process has tunable and untunable parametric vectors. The best candidate of the tunable vector in one process is migrated into another process as untunable vector through a migration controller. The migration strategy is adaptively controlled by a first-order projection of the fitness value's limits in each generation. This can decentralize the computations that ensure its suitability and applicability for network-based optimization. Optimization of a buck regulator for satisfying steady state and transient requirements is illustrated. Simulation predictions are verified with experimental measurements.

2. Decoupling of PEC's

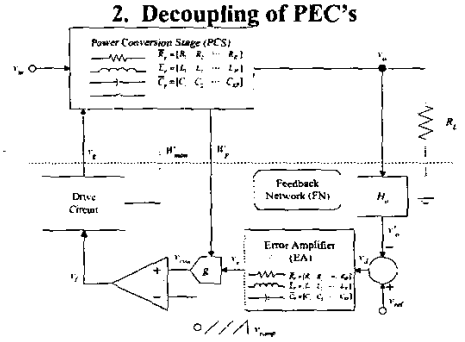


Fig. 1 Basic block diagram of a PEC.

0-7803-7804-0 /03/$17.00 © 2003 IEEE

The basic block diagram of a PEC including the PCS and FN is shown in Fig. 1. The PCS is supplied from the source v_{in} to the load R_L. 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 sent to an error amplifier (EA). The EA output v_e is combined with the feedback signals W_p 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 gate signals for the switches in the PCS. As defined below, vectors Θ_{PCS} and Θ_{FN} contain all passive component values for the PCS and FN, respectively,

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

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 $\overline{C}_F = [C_1 \quad C_2 \quad \cdots \quad C_{K_F}]$

3. Training Mechanism

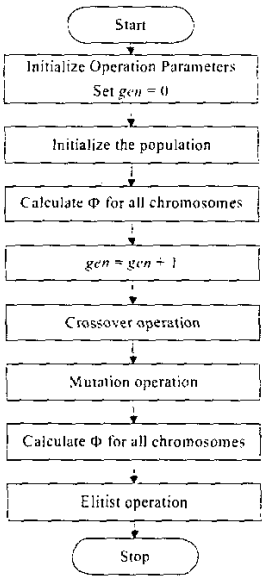
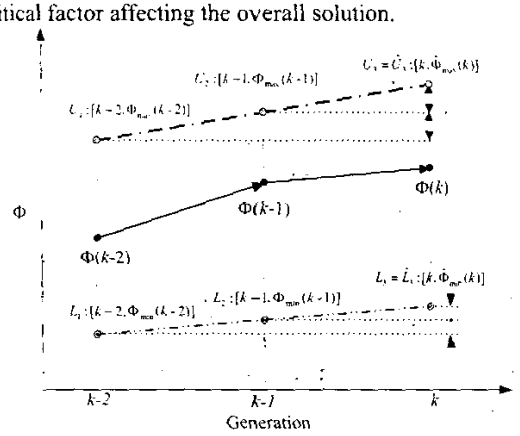


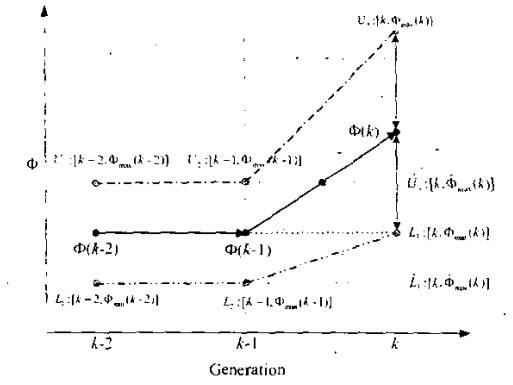
Fig. 2 Functional block diagram of the GA's.

All elements in Θ_{PCS} and Θ_{FN} are optimized through two training processes, namely Process I and Process II, respectively. Θ_{PCS} is a tunable vector and Θ_{FN} is an untunable one in Process I, whilst Θ_{PCS} is an untunable vector and Θ_{FN} is a tunable one in Process II. Each process utilizes classical GA's that inherit characteristics of evolutionary computations, involving randomness, recombination, and survival of the fittest. Detailed description of the GA's implementation can be found in [6] and is depicted in Fig. 2. Both processes have the same fitness function for showing the degree of attainment of the chromosome on the optimization objectives. The best candidate of the tunable vector in one process is migrated into the other process as untunable vector through a migration controller. Fig. 3 depicts an example of variations of the fitness value Φ of the best candidate versus the generation k. As the candidate trained in a generation must be no worse than the previous ones, Φ monotonically increases [i.e.$\Phi(k) \geq \Phi(k-1)$].

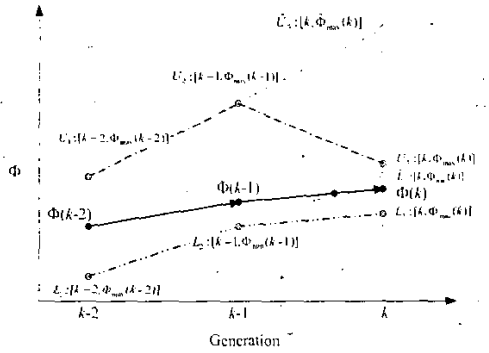
Since the migration frequency determines the effectiveness and efficiency in optimizing the circuit parameters, the proposed asynchronous migration scheme will adaptively adjust the migration frequency. The methodology is based on monitoring the change in Φ [i.e., $\Delta\Phi(k)$] between the kth and $(k-1)$th generations, where $\Delta\Phi(k) = \Phi(k) - \Phi(k-1)$. If $\Delta\Phi(k)$ is nonzero, a migration counter N^- will be incremented by one. If N^- equals a fixed reference value ν, migration will be initiated. The best candidate of the tunable vector in one process will be passed to the other process. In general, too frequent migration results in a single freely inter-bleeding population similar to classical GA's [8] [9]. Too little migration results in two diversified populations and two local optimization solutions. Hence, the value of ν is a critical factor affecting the overall solution.



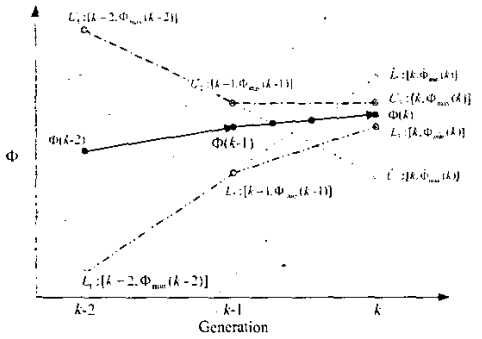
(a) $\hat{\Phi}_{max}(k) > \Phi(k) > \hat{\Phi}_{min}(k)$.



(b) $\Phi(k) > \hat{\Phi}_{max}(k)$.



(c) $\Phi(k) < \hat{\Phi}_{\min}(k)$.



(d) $\hat{\Phi}_{\max}(k) < \hat{\Phi}_{\min}(k)$.

Fig. 3 Illustrations on the variation of Φ against k.

A. Migration Controller

v is adjusted adaptively by comparing $\Phi(k)$ with the expected maximum bound $\Phi_{\max}(k)$ and minimum bound $\Phi_{\min}(k)$. As depicted in Fig. 3(a), the predicted upper bound $\hat{\Phi}_{max}(k)$ is formed by constructing a straight line passing through the point U_1: [k - 2, $\Phi_{\max}(k-2)$] and the point U_2: [k - 1, $\Phi_{\max}(k-1)$]. By using linear extrapolation, a line passing through U_1 and U_2 is formulated. $\hat{\Phi}_{\max}(k)$ (i.e., point $\hat{U}_3$) can be expressed as

$$\hat{\Phi}_{\max}(k) = \Phi_{\max}(k-1) + [\Phi_{\max}(k-1) - \Phi_{\max}(k-2)] = 2\Phi_{\max}(k-1) - \Phi_{\max}(k-2) \quad (2)$$

Similarly, the predicted lower bound $\hat{\Phi}_{\min}(k)$ (i.e., point $\hat{L}_3$) can be expressed as

$$\Phi_{\min}(k) = \Phi_{\min}(k-1) + [\Phi_{\min}(k-1) - \Phi_{\min}(k-2)] = 2\Phi_{\min}(k-1) - \Phi_{\min}(k-2) \quad (3)$$

As shown in Fig. 3(a), if $\hat{\Phi}_{\max}(k) > \Phi(k) > \hat{\Phi}_{\min}(k)$,

$$\Phi_{\max}(k) = \hat{\Phi}_{\max}(k) \quad (4)$$

and

$$\Phi_{\min}(k) = \hat{\Phi}_{\min}(k) \quad (5)$$

Otherwise, new bounds $\Phi_{\max}'(k)$ and $\Phi_{\min}'(k)$ for $\Phi_{\max}(k)$ and $\Phi_{\min}(k)$, respectively, will be formulated by linear interpolation between generations k and k - 1, where

$$\Phi_{\max}'(k) = \Phi(k) + \Delta\Phi(k) = 2\Phi(k) - \Phi(k-1) \quad (6)$$

$$\Phi_{min}'(k) = \Phi(k) - \Delta\Phi(k-1) = \Phi(k-1) \quad (7)$$

For example, as shown in Fig. 3(b), $\Phi(k) > \hat{\Phi}_{\max}(k)$ and is outside $\hat{U}_3$ and $\hat{L}_3$. New bounds U_3 and L_3 are derived from (6) and (7), respectively. The situation will be the same for $\Phi(k) < \hat{\Phi}_{\min}(k)$, which is illustrated in Fig. 3(c).

However, as shown in Fig. 3(d), it is possible that $\hat{\Phi}_{\max}(k) < \hat{\Phi}_{\min}(k)$. Concluding the above cases, $\Phi_{\max}(k)$ and $\Phi_{\min}(k)$ are generally determined by the following equations

$$\Phi_{\max}(k) = \max[\hat{\Phi}_{\max}(k), \Phi_{\max}'(k)] \quad (8)$$

and

$$\Phi_{min}(k) = \min[\hat{\Phi}_{min}(k), \Phi_{min}'(k)] \quad (9)$$

Based on the above steps, the following cases occur and have the adjustments on $\Phi_{\max}(k)$, $\Phi_{\min}(k)$, and v. The flowchart is shown in Fig. 4.

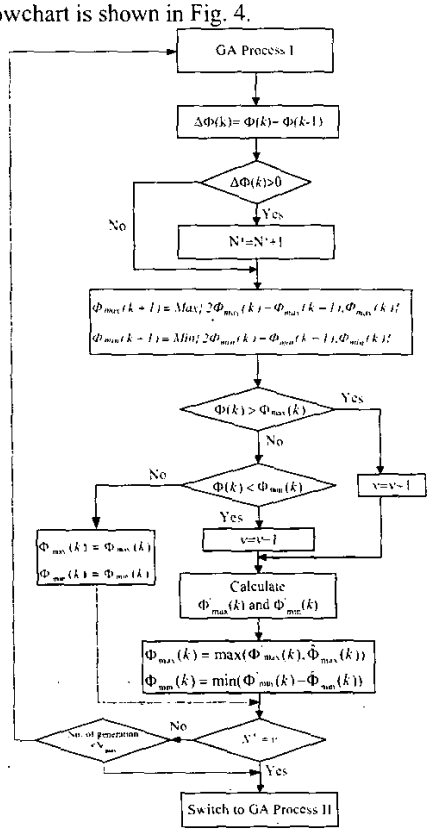


Fig. 4 Flowchart of the training mechanism.

Case 1 – If $\Phi_{\min}(k) < \Phi(k) < \Phi_{\max}(k)$, the value of ν is kept constant. If $\Delta\Phi(k+1)$ is positive, N^- is increased by one. Migration occurs when $N^- = \nu$.

Case 2 – If $\Phi(k) > \hat{\Phi}_{\max}(k)$, the untunable parameters due to the last migration show a high degree of collaboration with the tunable parameters of the current process. The tunable parameters of the current process are likely to have further optimization. The migration process has to be deferred by one generation (i.e., $\nu = \nu + 1$). Eqs. (8) and (9) will be applied.

Case 3 – If $\Phi(k) < \hat{\Phi}_{\min}(k)$, the untunable parameters due to the last migration show a low degree of collaboration with the tunable parameters of the current process. Thus, the tunable parameters of the current process have low potential for further optimization. The migration process has to be accelerated by one generation (i.e., $\nu = \nu - 1$). Again, the new bounds will be determined by (8) and (9), respectively.

B. *Training Algorithm*

With the help of Fig. 5, the training algorithm is summarized as below. The shaded blocks Θ_{FN} in Process I and Θ_{PCS} in Process II are untunable vectors. The two processes are started and executed as classical GA's.

Step 1 - Calculate the fitness values $\Phi(k)$ and $\Delta\Phi(k)$ of process I. Update the population of Process I by using standard GA's [6]. If $\Delta\Phi(k) > 0$, $N^- = N^- + 1$. Otherwise, N^- is kept constant.

Step 2 - Calculate $\hat{\Phi}_{\max}(k)$ and $\hat{\Phi}_{\min}(k)$ by using (2) and (3), respectively.

Step 3 - If $\Phi(k) > \hat{\Phi}_{\max}(k)$, $\nu = \nu + 1$. Update $\Phi_{\min}(k)$ and $\Phi_{\max}(k)$ by (8) and (9), respectively.

Step 4 - If $\Phi(k) < \hat{\Phi}_{\min}(k)$, $\nu = \nu - 1$. Update $\Phi_{\min}(k)$ and $\Phi_{\max}(k)$ by (8) and (9), respectively.

Step 5 - If $\hat{\Phi}_{\min}(k) < \Phi(k) < \hat{\Phi}_{\max}(k)$, $\Phi_{\max}(k) = \hat{\Phi}_{\max}(k)$ and $\Phi_{\min}(k) = \hat{\Phi}_{\min}(k)$.

Step 6 - If $N^- < \nu$, repeat steps 1 to 5 for Process I. If $N^- = \nu$, the best candidate (i.e., the tunable vector) in Process I is ready to migrate to Process II. Process I will wait until Process II becomes ready for migration. It should be noted that step 1) to step 6) are also performed in Process II.

Step 7 - Migration is initiated if both processes are ready for migration. Θ_{PCS} in Process I will be passed to Process II as untunable parameters, whilst Θ_{FN} in Process II will be passed to Process I as untunable parameters.

Step 8 - After migration, reset N^- to one. Repeat steps 1 to 8 for both processes.

It can be seen that ν is adjusted in an adaptive way throughout the overall training process. ν acts as an adaptive reference for controlling the migration process. The migration frequency is increased when there is no potential for further optimization in individual GA process and the frequency is reduced decrease when there is further potential for optimization in individual GA process.

As illustrated in Fig. 5, at the ith generation in Process I, $N^- = \nu$. Process I is ready for migration. However, as Process II is not ready for migration, Process I is kept waiting until at the kth generation of Process II. Migration is initiated. Θ_{PCS} in Process I is passed to Process II (white arrow). Θ_{FN} in Process II is passed to Process I (black arrow). Similarly, at the mth generation of Process II, Process II is ready for migration but Process I is not. Process II is then kept waiting until at the nth generation of Process I.

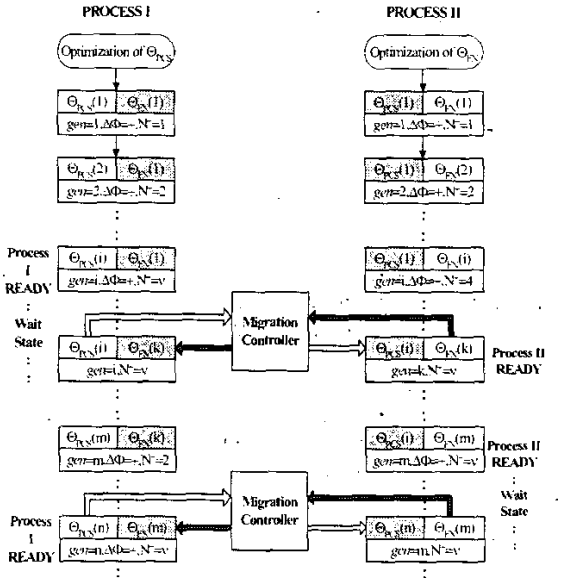


Fig. 5 Illustration of the migration process.

Finally, if migrations have not occurred for a number of generations $N_{\max}$, a forced migration is initiated. This is to prevent the two processes trapping in their corresponding local solutions and being acted as two isolated non-interactive GA processes. Moreover, the solutions will be accepted as the final ones if no self-migrations are initiated for more than N_{SM} times. In this situation the two GA processes come to the final value. No further optimization and searching for Θ_{PCS} and Θ_{FN} will be performed.

C. *Pseudo-Coevolutionary Implementation*

The parallel computations can be realized by a so-called pseudo-coevolutionary method, in which sequential computations are implemented. As shown in Fig. 5, Process I is firstly carried out from Step 1 to Step 6 until $N^- = \nu$. Process I will be paused and Process II will be initiated and proceeded from Step 1 to Step 6 until $N^- = \nu$. Finally, both processes are ready for migration. The best candidate of Θ_{PCS} in Process I will be passed to Process II and the best candidate of Θ_{FN} in Process II will be passed to Process I. The training will be continued for next computation.

D. *Fitness functions*

A fitness value is assigned to each chromosome *ch* in the population according to a predefined fitness

function Φ. The fitness value shows the degree of attainment of the chromosome on the optimization objectives. In this paper, a multi-objective optimization for optimizing the chromosome is adopted. Their definitions are described as follows.

1) steady state error of v_o,
2) maximum overshoot and undershoot, damping ratio, and the settling time of v_d during disturbances,
3) steady-state ripple voltage on v_o, and
4) dynamic behaviors.

It should be noted that other objective functions, such as cost function, can also be included [11]. Φ is defined as

$$\Phi(ch) = [\sum_{R_L = R_{L,min}, \delta R_L}^{R_{L,max}} \sum_{v_{in} = V_{in,min}, \delta v_{in}}^{V_{in,max}} OF_1(R_L, v_{in}, ch) + OF_2(R_L, v_{in}, ch) + OF_3(R_L, v_{in}, ch)] + OF_4(ch) \quad (10)$$

R_L varies from $R_{L,min}$ to $R_{L,max}$ and v_{in} varies from $V_{in,min}$ to $V_{in,max}$.

1. OF_1 for objective (1)

This objective is to evaluate the steady performance of the regulator. In order to measure the steady state performance, the steady state value of control signal v_{con} is determined by a dual loop iterative method in [9]. The filtered output v_o' is compared with the reference value v_{ref}. An integral square error function E_2 is defined to estimate the closeness of v_o' with v_{ref} in the N_s simulated samples. If no steady-state conditions can be found, OF_1 should be small. Otherwise, OF_1 should be large. Hence,

$$OF_1 = K_1 \, e^{-E_2/(K_2 \, E)} \quad (11)$$

where K_1 and K_2 are scaling factors. K_1 is the maximum attainable index of OF_1 and K_2 adjusts the sensitivity of OF_1 with respect to E_2. OF_1 decreases as E_2 increases.

2. OF_2 and OF_4 for objective (2) and objective (4)

During the startup or external disturbances, a transient response appears at v_d, where

$$v_d = v_{ref} - v_o' \quad (12)$$

OF_2 and OF_4 are used to measure the transient response of v_d, including 1) the maximum overshoot, 2) the maximum undershoot, and 3) the settling time of the response, during the startup and disturbances, respectively. The general form of OF_2 and OF_4 can be expressed as

$$OF_2 = OV(R_L, v_{in}, ch) + UV(R_L, v_{in}, ch) + ST(R_L, v_{in}, ch) \quad (13)$$

$$OF_4 = \sum_{i=1}^{N_T} OV(R_{L,i}, v_{in,i}, ch) + UV(R_{L,i}, v_{in,i}, ch) + ST(R_{L,i}, v_{in,i}, ch) \quad (14)$$

where N_T is the number of the input and load disturbances in the performance test.

In the above expressions, OV, UV, and ST are the objective functions for minimizing the maximum overshoot, maximum undershoot, and settling time of v_d. They are defined as,

$$OV = \frac{K_3}{1 + e^{-[(M_{p0} - M_p)/v_{ref}]/K_4}} \quad (15)$$

where K_3 is the maximum attainable value of this objective function, M_{p0} is the desired maximum overshoot, M_p is the actual overshoot, and K_4 is the passband constant.

$$UV = \frac{K_5}{1 + e^{-[(M_{v0} - M_v)/v_{ref}]/K_6}} \quad (16)$$

where K_5 is the maximum attainable value of this objective function, M_{v0} is the desired maximum undershoot, M_v is the actual undershoot, and K_6 is the passband constant.

$$ST = \frac{K_7}{1 + e^{-(T_{S0} - T_S)/K_8}} \quad (17)$$

where K_7 is the maximum attainable value of the objective function, T_{s0} is the desired settling time, T_s is the actual settling time, and K_8 is the passband constant. T_S is defined as the settling time of v_d that falls within a $\pm\sigma$ % band. That is,

$$|v_d(t)| \le 0.01\,\sigma, \; t \ge T_S \quad (18)$$

3. OF_3 of objective (3)

The objective function OF_3 is defined for measuring the maximum ripple voltage at v_o in steady state. In practice, the steady state ripple has to be less than a limit of $\pm\Delta v_o$ around the expected output $v_{o,exp}$. A measure of the attainment is to count the number of samples N_o that are outside $v_{o,exp} \pm \Delta v_o$ in a total number of N_s samples. OF_3 is defined as

$$OF_3 = K_9 \, (1 - \frac{N_o}{N_s}) \quad (19)$$

where K_9 is the scaling factor, which is maximum attainable index for this objective. OF_3 decreases as N_o increases.

4. Design Example

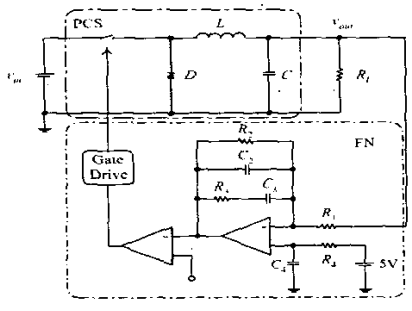


Fig. 6 Circuit schematic of the buck regulator with overcurrent protection.

The proposed method is illustrated with the same example in [11]. The schematic is shown in Fig. 6. The PCS is a classical buck converter and the FN is a

proportional-plus-integral controller. In [11], the component values of the PCS is determined by considering the steady-state behaviors of the buck converter whilst the FN is determined by considering the overall performance of the whole circuit. Although the method requires a shorter computation time than the overall training method, there is no interaction between the PCS and FN in training the PCS. The trained parameters in [11] are tabulated in Table I, whilst the ones with the proposed optimization method are tabulated in Table II.

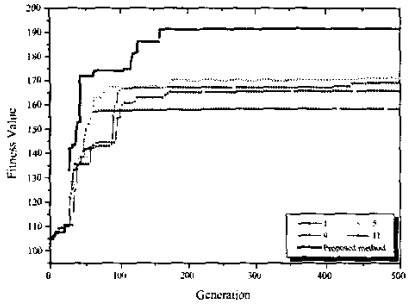
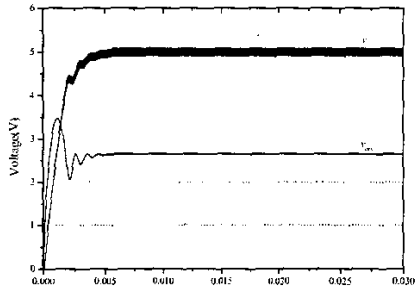
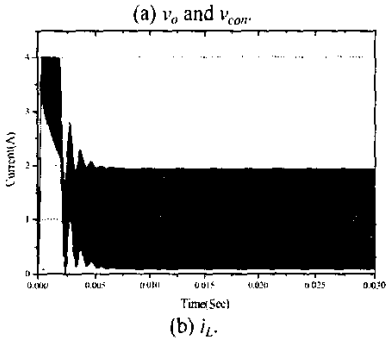


Fig. 7Comparisons of the fitness values against the training generations with fixed and the proposed migration strategies.

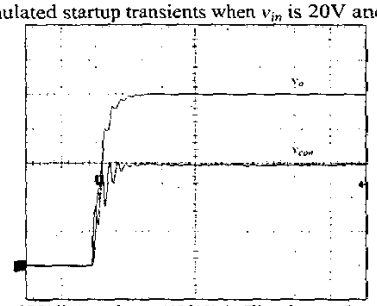


(a) v_o and v_{con}.

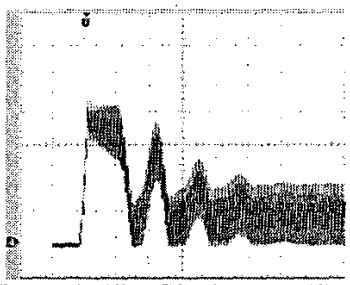


(b) i_L.

Fig. 8 Simulated startup transients when v_{in} is 20V and R_L is 5Ω.

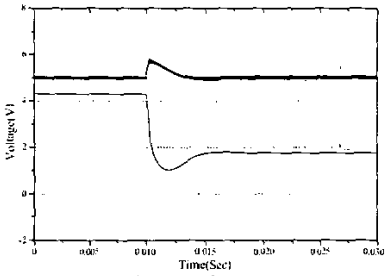


(a) v_o (1V/div) and v_{con} (1V/div) (Timebase: 5ms/div).

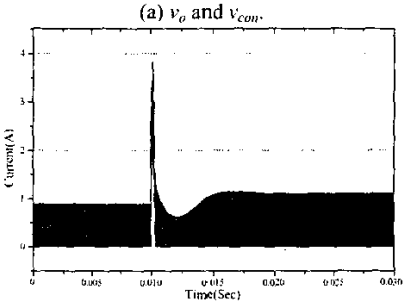


(b) i_L (1.0A/div) (Timebase: 1ms/div).

Fig. 9 Experimental startup transients when v_{in} is 20V and R_L is 5Ω.

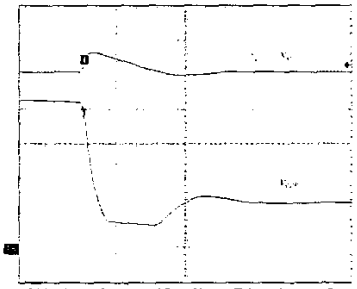


(a) v_o and v_{con}.

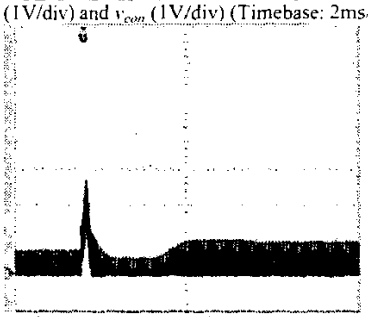


(b) i_L.

Fig. 10 Simulated transient responses when v_{in} is changed from 10V into 20V.

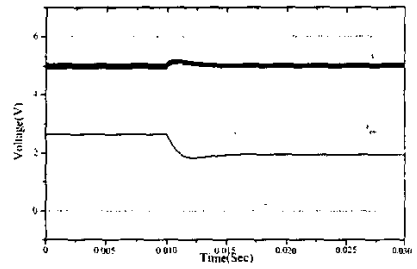


(a) v_o (1V/div) and v_{con} (1V/div) (Timebase: 2ms/div).

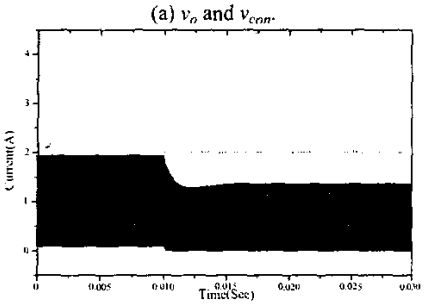


(b) i_L (1A/div) (Timebase: 2ms/div).

Fig. 11 Experimental transient responses when v_{in} is changed from 10V into 20V.

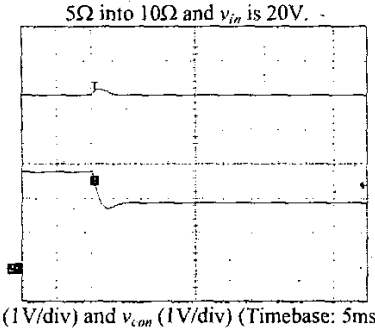


(a) v_o and v_{con}.

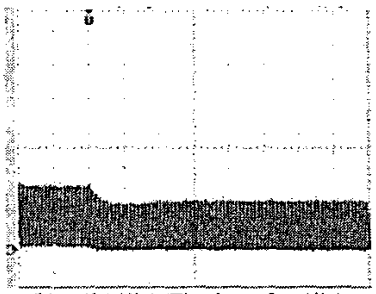


(b) i_L.

Fig. 12 Simulated transient responses when R_L is changed from 5Ω into 10Ω and v_{in} is 20V.



(a) v_o (1V/div) and v_{con} (1V/div) (Timebase: 5ms/div).



(b) i_L (1A/div) (Timebase: 2ms/div).

Fig. 13 Experimental transient responses when R_L is changed from 5Ω into 10Ω and v_{in} is 20V. (Timebase: 2ms/div).

Table I Optimized parameters in [11]

(a) Initial values of L and C and the results after 500 generations.

Component	Initial Value	Optimized value after 500 generations
1 L	200μH	194μH
2 C	1000μF	1054μF

(b) Initial component values for the controller and the results after 500 generations.

Component	Initial Value	Optimal Value after 500 generations
R_{C3}	4.7kΩ	3.5 kΩ
C_2	2μF	5.9 μF
C_3	3.3μF	0.46 μF
R_2	300kΩ	767 kΩ
C_4	1.8μF	1.1 μF
R_4	1kΩ	6.5 kΩ
R_1	0.6kΩ	1.1 kΩ

Table II Optimized parameters with the proposed method.

(a) Initial values of L and C and the results after 500 generations.

Component	Initial Value	Optimized value after 500 generations
L	200μH	105μH
C	1000μF	1196μF

(b) Initial component values for the controller and the results after 500 generations.

Component	Initial Value	Optimal Value after 500 generations
R_{C3}	4.7kΩ	0.749 kΩ
C_2	2μF	0.24 μF
C_3	3.3μF	1.865 μF
R_2	300kΩ	1039.8 kΩ
C_4	1.8μF	8.0 μF
R_4	1kΩ	0.179 kΩ
R_1	0.6kΩ	0.208 kΩ

(c) Parameters used in the optimization.

Parameter	Value	Parameter	Value
p_x	0.85	K_5	2
P_m	0.25	K_6	0.455
G_{max}	500	K_7	2
N_p	30	K_8	2.28×10^{-3}
N_s	15000	K_9	2

N_T	6	τ_L	4.55×10^{-3}
K_1	2	τ_C	2.14×10^{-3}
K_2	400	δv_{in}	20V
K_3	2	δR_L	3 Ω
K_4	0.455		

Fig. 7 shows comparisons of the fitness values against the training generations with fixed migration frequency and the proposed adaptive migration frequency. As expected, for the fixed migration frequency scheme, there is a critical value that can optimize the components. A value of five is the best one in this example. Higher or lower than five, such as one, nine, and eleven shown in the figure, cannot enhance the optimization problem. The proposed method is unnecessary to preset the migration frequency and is adaptively adjusted. The convergence rate and the final fitness value are better than the fixed migration rate, because interaction in optimizing the PCS and FN is enhanced.

Fig. 8 and Fig. 9 shows the simulated experimental startup transients, when the input voltage is 20V and the output load resistance is 5Ω. The settling time is less than 10ms, which is shorter than the design in [11]. Figs. 10 and 11 show the theoretical and experimental transients when the input voltage is changed from 10V to 20V. Figs. 12 and 13 show the theoretical and experimental transients when the output load is changed from 5Ω to 10Ω. The experimental measurements are in close agreement with the predicted results. 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 [11], confirming the advantage of the proposed method.

5. Conclusions

Pseudo-coevolutionary genetic algorithms for design of power electronic circuits have been presented. The two parallel optimization processes entail selection of the component values in the PCS and FN, in order to meet the required steady state and dynamic performances. An adaptive migration mechanism that determines the exchange rate has been proposed. Detailed implementation procedures have been described. Illustrative example shows that the optimized values give higher fitness value than the method with fixed migration and the decoupled optimization technique in [11]. It is because interactions between the PCS and FN in the optimization are improved.

References

[1]. V. Petridis, S. Kazarlis, and A. Bakirtzis, "Varying fitness functions in genetic algorithm constrained optimization: The cutting stock and unit commitment problems," *IEEE Trans. System, Man and Cybernetics B*, vol.28, no.5, pp. 629-640, Oct. 1998.

[2].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. Evolutionary Computation*, 1998, pp. 301-305.

[3].M. Wojcikowski, J. Glinianowicz, and M. Bialko, "System for optimisation of electronic circuits using genetic algorithm," in *Proc. IEEE Int. Conf. Electronics, Circuits, and Systems*, 1996, vol. 1, pp. 247-250.

[4].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. Sym. Circuits Systs.*, 1999, vol. 6, pp. 362-265.

[5].Z. Michalewicz, *Genetic algorithms + Data Structure = Evolution Programs*, Springer-Verlag, 1996.

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

[7].E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Parallels, Reseaux et Systems Repartis*, Paris: Hermes, vol. 10, no. 2, pp. 141-171.

[8].M. Potter and K. De Jong, "Cooperative coevolution: an architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1-29, 2000.

[9].M. Potter, and K. De Jong, "A cooperative coevolutionary approach to function optimisation.", Proceedings of the third conference on Parallel problem solving form Nature, pp. 245-257, Springer-Verlag, 1994.

[10]. B. Wong and H. Chung, "Steady-state analysis of PWM dc/dc switching regulators using iterative cycle time-domain simulation," *IEEE Trans. Ind. Electron.*, vol. 45, no. 3, pp. 421-432, June 1998.

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