Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 6519678, 19 pages
http://dx.doi.org/10.1155/2016/6519678

Research Article

A Chaos-Enhanced Particle Swarm Optimization with Adaptive Parameters and Its Application in Maximum Power Point Tracking

Ying-Yi Hong,[1] Angelo A. Beltran Jr.,[2,3] and Arnold C. Paglinawan[3,4]

[1]*Department of Electrical Engineering, Chung Yuan Christian University, Chung Li District, Taoyuan City 320, Taiwan*
[2]*Center for Research & Development and Department of Electronics Engineering, Adamson University, 1000 Manila, Philippines*
[3]*School of Graduate Studies, Mapua Institute of Technology, 1002 Manila, Philippines*
[4]*School of Electrical Electronics Computer Engineering, Mapua Institute of Technology, 1002 Manila, Philippines*

Correspondence should be addressed to Ying-Yi Hong; yyh10632@yahoo.com.tw

Received 11 April 2016; Accepted 5 July 2016

Academic Editor: Zhen-Lai Han

This work proposes an enhanced particle swarm optimization scheme that improves upon the performance of the standard particle swarm optimization algorithm. The proposed algorithm is based on chaos search to solve the problems of stagnation, which is the problem of being trapped in a local optimum and with the risk of premature convergence. Type 1″ constriction is incorporated to help strengthen the stability and quality of convergence, and adaptive learning coefficients are utilized to intensify the exploitation and exploration search characteristics of the algorithm. Several well known benchmark functions are operated to verify the effectiveness of the proposed method. The test performance of the proposed method is compared with those of other popular population-based algorithms in the literature. Simulation results clearly demonstrate that the proposed method exhibits faster convergence, escapes local minima, and avoids premature convergence and stagnation in a high-dimensional problem space. The validity of the proposed PSO algorithm is demonstrated using a fuzzy logic-based maximum power point tracking control model for a standalone solar photovoltaic system.

1. Introduction

Swarm intelligence is becoming one of the hottest areas of research in the field of computational intelligence especially with regard to self-organizing and decentralized systems. Swarm intelligence simulates the behavior of human and animal populations. Several swarm intelligence optimization algorithms can be found in the literature, such as ant colony optimization, artificial bee colony optimization, the firefly algorithm, differential evolution, and others. These are biologically inspired optimization and computational techniques that are based on the social behaviors of fish, birds, and humans. Particle swarm optimization (PSO) is a nature-inspired algorithm that draws on the behavior of flocking birds, social interactions among humans, and the schooling of fish. In fish schooling, bird flocking, and human social interactions, the population is called a swarm and candidate solutions, corresponding to the individuals or members in the swarm, are called particles. Birds and fishes generally travel in a group without collision. Accordingly, using the group information for finding the shelter and food, each particle adjusts its corresponding position and velocity, representing a candidate solution. The position of a particle is influenced by neighbors and the best found solution by any particle. PSO is a population-based search technique that involves stochastic evolutionary optimization. It is originally developed in 1995 by Eberhart and Kennedy [1, 2] to optimize constrained and unconstrained, continuous nonlinear, and nondifferentiable multimodal functions [1, 3]. PSO is a metaheuristic algorithm that was inspired by the collaborative or swarming behavior of biological populations [4]. Since then, it has been applied to solve a wide range of optimization problems, such as constrained and unconstrained problems, multiobjective problems, problems

with multiple solutions, and optimization in dynamic environments [5–8]. Some of the advantages of particle swarm optimization are the following: (a) computational efficiency [6], (b) effective convergence and parameter selection [7], (c) simplicity, flexibility, robustness, and ease of implementations [9], (d) ability to hybridize with other algorithms [10], and many others. PSO has few parameters to adjust and a small memory requirement and uses few CPU resources, making it computationally efficient. Unlike simulated annealing which can work only with discrete variables, PSO can work for both discrete and analog variables without ADC or DAC conversion. Also, genetic algorithm optimization requires crossover, selection, and mutation operators, whereas PSO utilizes only the exchange of information among individuals searching the problem space repeatedly [11]. In recent years, the use of particle swarm optimization has been investigated with focus on its use to solve a wide range of scientific and engineering problems such as fault detection [12], parameter identification [13, 14], power systems [15–17], transportation [18], electronic circuit design [19], and plant control design [20]. Most relevant research focuses on either constrained or unconstrained optimization problems.

The particle swarm optimization was developed to optimally search for the local best and the global best; these searches are frequently known as the exploitation and exploration of the problem space, respectively. Hong et al. [21] stated that exploitation involves an intense search of particles in a local region while exploration is a long term search, whose main objective is to find the global optimum of the fitness function. Although particle swarm optimization rapidly searches the solution of many complex optimization problems, it suffers from premature convergence, trapping at a local minimum, the slowing down of convergence near the global optimum, and stagnation in a particular region of the problem space especially in a multimodal functions and high-dimensional problem space. If a particle is located at the position of the global best and the preceding velocity and weight inertia are non-zero, then the particle is moving away from that particular point [16, 22]. Premature convergence happens if no particle moves and the previous velocities are near to zero. Stagnation thus occurs if the majority of particles are concentrated at the best position that is disclosed by the neighbors or the swarm. This fact has in recent years motivated various investigations by several researchers on variants of the particle swarm optimization, in an attempt to improve the performance of exploitation and exploration and to eliminate the aforementioned problems. The various methods of particle swarm optimization have been used for several purposes, including scheduling, classification, feature selection, and optimization.

Mendes et al. [23] presented fully informed particle swarm optimization, in which, during the optimization search, particles are influenced by the best particles in their neighborhood and information is evenly distributed during the generations of the algorithm. Liang et al. [24] proposed a comprehensive learning PSO in which each particle learns from the other neighborhood personal best at different dimensions. Accordingly, particles update their velocity based on the history of the their own personal bests. Wang et al. [25, 26] developed the opposition-based PSO with Cauchy mutation. Their technique uses an opposition learning scheme in which the Cauchy mutation operator helps the particles move to the best positions. Pant et al. [27] modified the inertia weight to exhibit a Gaussian distribution. Xiang et al. [28] applied the time delay concept PSO to enable the processing of information by particles to find the global best. Cui et al. [29] presented the fitness uniform selection strategy (FUSS) and the random walk strategy (RWS) to enhance the exploitation and exploration capabilities of PSO. Montes de Oca et al. [30] developed Frankenstein's PSO, which incorporates several variants of PSO in the literature such as constriction [31], the time-varying inertia weight optimizer [32, 33], the fully informed particle swarm optimizer [23], and the adaptive hierarchical PSO [34]. The adaptive PSO that was proposed by Zhan et al. [35] utilized the information that was obtained from the population distribution and fitness of particles to determine the status of the swarm and an elitist learning strategy to search for the global optimum. Juang et al. [36] presented the use of fuzzy set theory to tune automatically the acceleration coefficients in the standard PSO. A quadratic interpolation and crossover operator is also incorporated to improve the global search ability. The literature includes hybridization of particle swarm optimization with other stochastic or evolutionary techniques [10, 37–39] to realize all of their strengths.

Every modification of the particle swarm optimization uses a different method to solve optimization problems. The investigations cited above therefore elucidate some improvements of the standard particle swarm optimization. However, variants of particle swarm optimization generally exhibit the following limitations. (a) The particles may be positioned in a region that has a lower quality index than previously, leading to a risk of premature convergence, trapping in local optima, and the impossibility of further improvement of the best positions of the particles because the inertia weight, cognitive factors, and social learning factors in the algorithm are not adaptive or self-organizing. (b) The inclusion of the mutation operator may improve the speed of convergence. Nevertheless, global convergence is not guaranteed because the method is likely to become trapped in the local optimum during local searches of several functions. (c) The probability in the algorithm may improve the updated positions of particles. However, the changes in the new positions of particles, consistent with the probabilistic calculations, can move the particles into the worst positions. (d) Improving information sharing and the particle learning process capability of the algorithm can provide several benefits, but doing so often increases CPU times for computing the global optimum. (e) Integrating particle swarm optimization with other evolutionary or stochastic algorithms may increase the number of required generations, the complexity of the algorithm, and the number of required calculations.

This paper proposes a novel particle swarm optimization framework. The primary merits of the proposed variant of particle swarm optimization are as follows. (a) A modified sine chaos inertia weight operator is introduced, overcoming the drawback of trapping in a local minimum which is

commonly associated with an inertia weight operator. Chaos search improves the best positions of the particles, favors rapid finding of solutions in the problem space, and avoids the risk of premature convergence. (b) Type $1''$ constriction coefficient [40] is incorporated to increase the convergence rate and stability of the particle swarm optimization. (c) Self-organizing, adaptive cognitive, and social learning coefficients [41] are integrated to improve the exploitation and exploration search of the particle swarm optimization algorithm. (d) The proposed optimization algorithm has simple structure reducing the required memory demands and the computational burden on the CPU. It can therefore easily be realized using a few, low-cost test modules.

The remainder of this paper is organized as follows. Section 2 presents the standard particle swarm optimization algorithm. Section 3 describes the proposed variant of particle swarm optimization algorithm. Section 4 discusses the performance of the proposed variant of the particle swarm optimization and compares results obtained when well known optimization methods are applied to benchmark functions. The proposed variant of particle swarm optimization is further utilized in maximum power point tracking control using fuzzy logic for a standalone photovoltaic system. Finally, a brief conclusion is drawn.

2. Standard Particle Swarm Optimization

The particle swarm optimization is a simulating algorithm, evolutionary, and a population-based stochastic optimization method that originates in animal behaviors such as the schooling of fish and the flocking of bird, as well as human behaviors. It has best position memory of all optimization methods and a few adjustable parameters and is easy to implement. The standard PSO does not use the gradient of an objective function and mutation [11]. Each particle randomly moves throughout the problem space, updating its position and velocity with the best values. Each particle represents a candidate solution to the problem and searches for the local or global optimum. Every particle retains a memory of the best position achieved so far, and it travels through the problem space adaptively. The personal best (p^t_{best}) is the best solution so far achieved by an individual in the swarm within the problem space $p^t_{\text{best}} = (p^t_{\text{best},1}, p^t_{\text{best},2}, p^t_{\text{best},3}, \ldots, p^t_{\text{best,pop}})$ while the global best (g_{best}) refers to globally obtained best solution by any particle within the swarm $g_{\text{best}} = (g_{i1}, g_{i2}, g_{i3}, \ldots, g_{id})$ in the problem space dimension d. The position and velocity of particle i in the problem space dimension d are thus given by $x_p(t) = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id})$ and $v_p(t) = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{id})$, respectively. The velocity and position of a particle p are adjusted as follows [1, 2]:

$$\begin{aligned} v_p^{t+1} &= \omega \times v_p^t + c_1 \times r_1 \times \left(p^t_{\text{best}} - x_p^t\right) + c_2 \times r_2 \\ &\quad \times \left(g_{\text{best}} - x_p^t\right) \\ x_p^{t+1} &= x_p^t + v_p^{t+1}, \end{aligned} \tag{1}$$

where the superscript t is the generation index, whereas c_1 and c_2 are cognitive and social parameters, which are frequently known as acceleration constants and which are mainly responsible for attracting the particles toward p^t_{best} and g_{best}. The terms r_1, r_2, and ω denote uniform random numbers $[r_1, r_2] \in [0, 1]$ and inertia weight $\omega \in [0, 1]$, respectively. These factors are mainly responsible for balancing the local and global optima search capabilities of the particles in problem space. Every generation, the velocity of individuals in the swarm is computed and which adjusted velocity is used to compute the next position of the particle. To determine whether the best solution is achieved and to evaluate the performance of each particle, the fitness function is included. The best position of each particle is relayed to all particles in the neighborhood. The velocity and the position of each particle are repeatedly adjusted until the halting criteria are satisfied or convergence is obtained.

3. Chaos-Enhanced Particle Swarm Optimization with Adaptive Parameters

This section demonstrates that the proposed variant of particle swarm optimization improves upon the performance of the standard particle swarm optimization consistent with (1). The novel scheme improves upon the performance of other population-based algorithms in solving high-dimensional or multimodal problems. Chaos operates in a nonlinear fashion and is associated with complex behavior, unpredictability, determinism, and high sensitivity to initial conditions. In chaos, a small perturbation in the initial conditions can produce dramatically different results [42, 43]. In 1963, Lorenz [44] presented an autonomous nonlinear differential equation that generated the first chaotic system. In recent years, the scientific community has paid increasing attention to the chaotic systems and their applications in various areas of science and engineering. Such systems have been investigated in such fields as parameter identifications [14], optimizations [45], electronic circuits [46], electric motor drives [47, 48], power electronics [49], communications [50], robotics [51], and many others.

Feng et al. [52] introduced two means of modifying the inertia weight of a PSO using chaos. The first type is the chaotic decreasing inertia weight and the second type is the chaotic random inertia weight. In this paper, the latter is considered intensifying the inertia weight parameter of the PSO. The dynamic chaos random inertia weight is used to ensure a balance between exploitation and exploration. A low inertia weight favors exploitation while a high inertia weight favors exploration. A static inertia weight influences the convergence rate of the algorithm and often leads to premature convergence. Chaotic search optimization in all instances was used herein because of its highly dynamic property, which ensures the diversity of the particles and escape from local optimum in the process of searching for the global optimum.

The logistic map $z_{t+1} = \mu z_t(1 - z_t)$ [53, 54], where $\mu = 4$ is a very common chaotic map, which is found in much of the literature on chaotic inertia weight; it does not guarantee chaos on initial values of $z_0 \notin \{0, 0.25, 0.5, 0.75, 1\}$ that may arise during the initial generation process. In this paper,

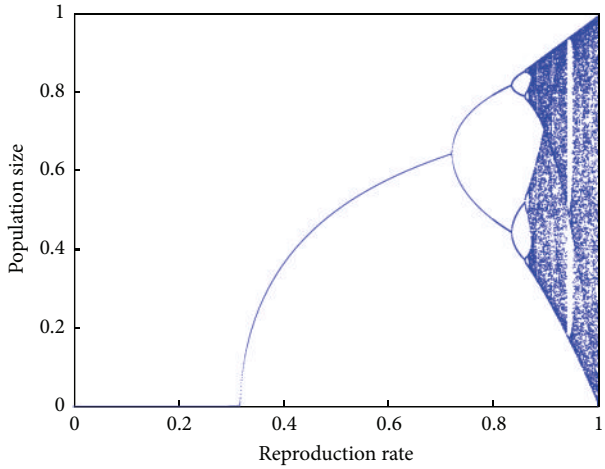

Figure 1: Bifurcations of sine map (z_{t+1}) at interval [0, 1].

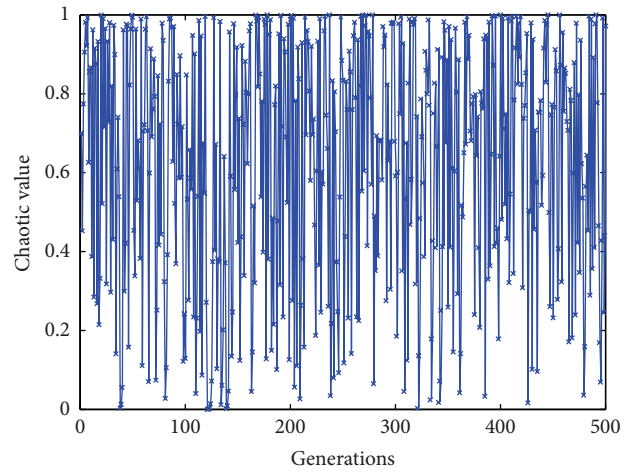

Figure 2: Chaotic value (z_{t+1}) with $z_0 = 0.7$ obtained using modified sine map (3) after 500 generations.

the sine chaotic map [54] given by (2) was utilized to avoid this shortcoming. Its simplicity eliminates complex calculations, reducing the CPU time:

$$z_{t+1} = \beta \sin(\pi z_t), \tag{2}$$

where $\beta > 0$, $z_t, z_{t+1} \in [0, 1]$ and t is the generation number. Figure 1 presents the bifurcation diagram of the sine chaotic map. In some instances of generations, z_{t+1} has relatively very small values. Hence, to improve the effectiveness of the chaos random inertia weight of particle swarm optimization, the original sine chaotic map is lightly modified as follows:

$$z_{t+1} = \left| \sin\left(\frac{\pi z_t}{\text{rand}(\cdot)} \right) \right|, \tag{3}$$

where $\beta = 1$ and $z_t, z_{t+1} \in [0, 1]$; the absolute sign ensures that the next-generation process in chaos space has $z_{t+1} \in [0, 1]$. Therefore, the chaotic random inertia weight ω^t_{chaos} is given by

$$\omega^t_{\text{chaos}} = 0.5 \times \text{rand}(\cdot) + 0.5 \times z_{t+1}. \tag{4}$$

Figure 2 plots the dynamics of the modified sine chaotic map while Figure 3 displays the bifurcation diagram.

Type 1″ constriction coefficient is integrated to the proposed variant of PSO to prevent the divergence of the particles during the search for solutions in problem space. The coefficient is used to fine-tune the convergence of particle swarm optimization. Consider

$$\chi = \frac{2}{\left(\phi - 2 + \sqrt{\phi^2 - 4\phi} \right)}, \tag{5}$$

where the parameter $\phi = c_1 + c_2$ depends on the cognitive and social parameters and the criterion $\phi > 4$ guarantees the effectiveness of the constriction coefficient. Incorporating the above coefficient ensures the quality of convergence and the stability of the generation process for particle swarm optimization.

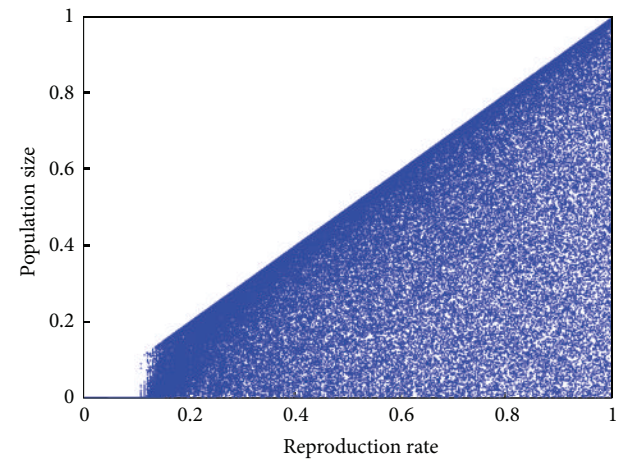

Figure 3: Bifurcations of modified sine map (z_{t+1}) at interval [0, 1].

Time-varying cognitive and social parameters are incorporated into PSO to improve its local and the global search by making the cognitive component large and the social component small at the initialization or in the early part of the evolutionary process. A linearly decreasing cognitive component and a linearly increasing social component in the evolutionary process enhance the exploitation and exploration of the PSO, helping the particle swarm to converge at the global optimum. The mathematical equation is represented as follows:

$$\begin{aligned} c_1^t &= c_{1,f} - \frac{t}{\text{MAXITR}} \left(c_{1,f} - c_{1,i} \right) \\ c_2^t &= c_{2,i} + \frac{t}{\text{MAXITR}} \left(c_{2,f} - c_{2,i} \right), \end{aligned} \tag{6}$$

where $c_{1,i}, c_{2,i}, c_{1,f}$, and $c_{2,f}$ are the initial and final values of the cognitive parameters and the social parameters, respectively; t is the current generation, and the MAXITR is the value in the final generation.

Table 1: Performance of different methods in minimizing sphere function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$7.4310E-14$	$1.1579E-11$	$3.1532E-08$	$1.1929E-07$	$1.2010E-08$
	Time	1.2669	1.4928	7.9243	7.3712	1.2056
Mean	Fitness	$4.60054E-13$	$3.3756E-07$	$4.30642E-08$	$3.46940E-07$	$2.78219E-08$
	Time	1.2845	1.4131	8.0510	7.3559	1.2117
Worst	Fitness	$1.2253E-12$	$2.3454E-06$	$5.7355E-08$	$8.8534E-07$	$5.4930E-08$
	Time	1.2849	1.4669	8.0826	7.4529	1.1936

Table 2: Shortest CPU times of different methods in minimizing sphere function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$7.4310E-14$	$9.0425E-07$	$3.1532E-08$	$3.7020E-07$	$4.0916E-08$
	Time	1.2669	1.3332	7.9243	7.2014	1.1381
Worst	Fitness	$2.4087E-13$	$1.1579E-11$	$4.8078E-08$	$1.8604E-07$	$2.8446E-08$
	Time	1.3084	1.4928	8.2801	7.4778	1.2438

The above components improve the performance of the standard PSO. Therefore, the proposed mathematical equation for the velocity and position of the particle swarm optimization are as follows:

$$v_p^{t+1} = \chi \times \omega_{\text{chaos}}^t \times v_p^t + c_1^t \times \left(p_{\text{best}}^t - x_p^t\right) + c_2^t \times \left(g_{\text{best}}^t - x_p^t\right) \quad (7)$$
$$x_p^{t+1} = x_p^t + v_p^{t+1}.$$

The uniform random numbers $[r_1, r_2] \in [0, 1]$ from the velocity equation of the standard PSO are not included in the proposed PSO. Figure 4 displays the flowchart of the proposed chaos-enhanced PSO.

The mathematical representations and algorithmic steps represent a significant improvement on the performance of the standard PSO. A numerical benchmark test was carried out using the unimodal and multimodal functions. The following section presents and discusses the results.

4. Simulation Results

In this section, four benchmark test functions are used to test the performance of the proposed algorithm. Subsections elucidate the names of the benchmark test functions, their search spaces, their mathematical representations, and variable domains.

Five programming codes were developed in Matlab 7.12 to minimize the above benchmark functions. These codes correspond to the proposed PSO, the standard PSO, the firefly algorithm (FA) [55, 56], ant colony optimization (ACO) [57, 58], and differential evolution (DE) [59, 60], which are evaluated using the benchmark test functions for comparative purposes.

The parameter settings for the above algorithms are as follows. For the PSO, the inertia weight ω, cognitive learning c_1, and social learning c_2 factors are given as 0.99, 1.5, and 2.0, respectively. For the FA, the light absorption γ, attraction β, and mutation α coefficients are 1.0, 2.0, and 0.2, respectively. For ACO, the selection pressure q and deviation-to-distance ratio ς are 0.5 and 1.0, respectively. The roulette wheel selection method is used for ACO. The mutation coefficient β and the crossover rate for DE are 0.8 and 0.2, respectively. The population size and the number of unknowns (dimensions) for all population-based algorithms that were used in the benchmark test are 20. Ten simulations tests are performed using each of the algorithms in order to evaluate their performance in minimization.

To verify the optimality and robustness of the algorithms, two convergence criteria are adopted: the convergence tolerance and the fixed maximum number of generations. The desktop computer that was used in the benchmark test function experiments ran the Microsoft Windows 7 64-Bit Operating System and had an Intel (R) Core *i*5 (3.30 GHz) processor with 8.0 GB RAM installed.

4.1. Benchmark Testing: Sphere Function. The sphere function is given by $f_1(x) = \sum_{i=1}^{d} x_i^2$, $-100 < x_i < 100$ where $d = 20$ and $x = (x_1, x_2, \ldots, x_{20})$. The sphere function $f_1(x)$ is a unimodal test function whose global optimum value is $f_1(x) = 0$ and $x_1 = x_2 = x_3 = \cdots = x_{20} = 0$. Table 1 presents the best, mean, and worst values obtained by running all the algorithms through 500 generations. Figure 5 shows the performance for the maximum number of generations of each population-based algorithms in minimizing $f_1(x)$. Table 2 presents the shortest CPU times that were required to minimize $f_1(x)$ under 500 generations and Figure 6 plots this information for all algorithms.

In the next experimental test, the convergence tolerance was set to 0.001. Table 3 presents the best, mean, and worst values that were used to minimize $f_1(x)$ using all techniques, based on ten simulation results. Almost all techniques provide similar solutions. As presented in Table 3, the proposed method gives smaller values of $f_1(x)$ in the fewest generations and in the shortest CPU times. Figure 7 shows the convergence performance in minimizing $f_1(x)$.

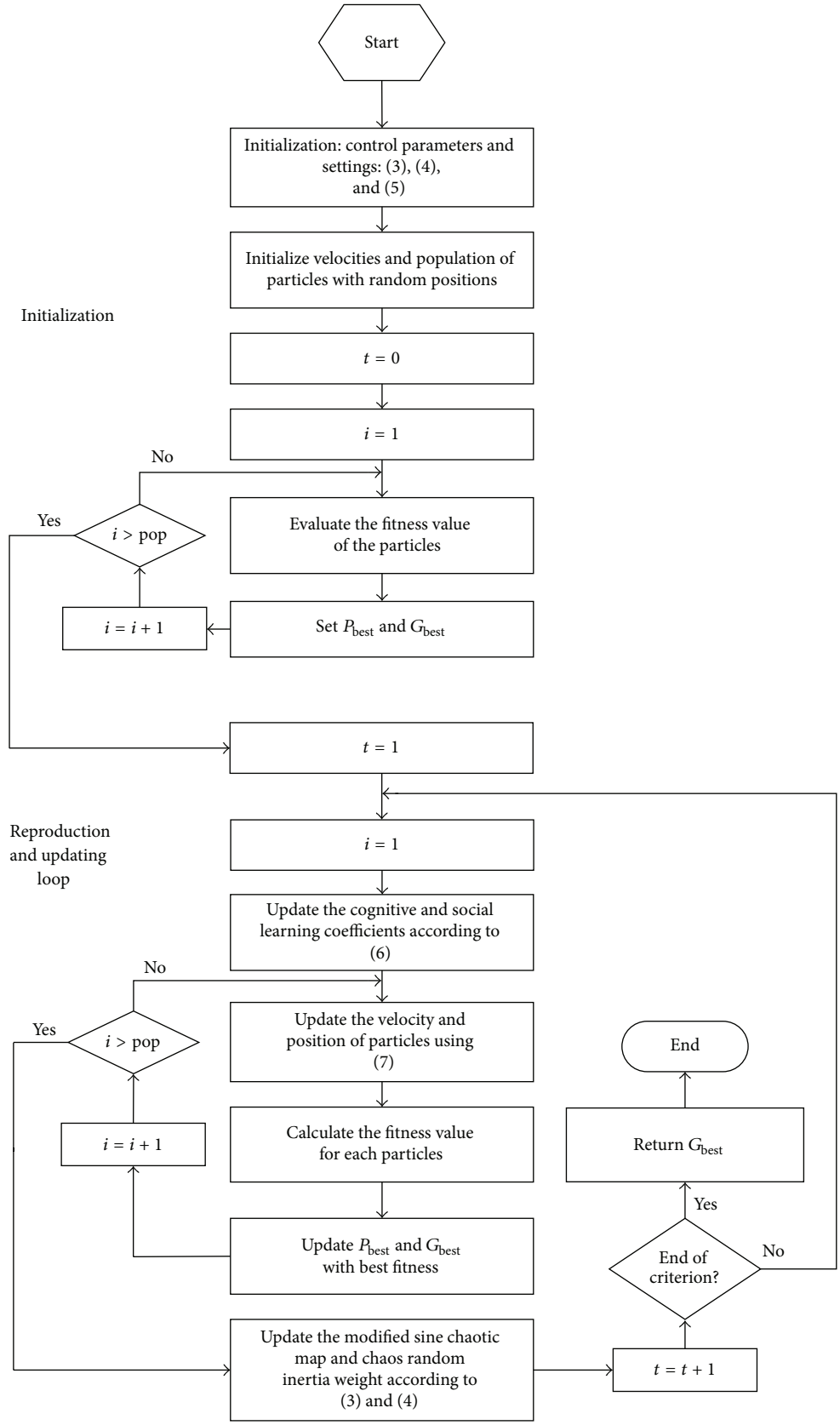

Figure 4: Flowchart of chaos-enhanced PSO with adaptive parameters.

Table 3: Performance of methods in minimizing sphere function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$6.8672E-04$	$4.3526E-04$	$7.0053E-04$	$5.6728E-04$	$6.5308E-04$
	Generations	162	191	252	354	306
	Time	0.4266	0.4942	4.1338	5.1278	0.7515
Mean	Fitness	$8.6914E-04$	$7.0088E-04$	$8.49667E-04$	$7.79853E-04$	$9.06594E-04$
	Generations	166.7	192.9	249.6	338.4	310.6
	Time	0.4418	0.5247	4.0568	4.9292	0.7725
Worst	Fitness	$9.95120E-04$	$9.2794E-04$	$9.4996E-04$	$9.9648E-04$	$9.9164E-04$
	Generations	168	199	253	338	301
	Time	0.4395	0.5376	4.1049	4.8423	0.7351

Table 4: Shortest CPU times for minimizing sphere function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$9.3883E-04$	$4.3526E-04$	$7.5282E-04$	$7.6357E-04$	$9.3375E-04$
	Generations	162	191	237	314	294
	Time	0.4188	0.4942	3.8547	4.6462	0.7171
Worst	Fitness	$8.0293E-04$	$9.2794E-04$	$8.4899E-04$	$9.7435E-04$	$8.8470E-04$
	Generations	164	199	260	353	331
	Time	0.4600	0.5376	4.1888	5.2340	0.9117

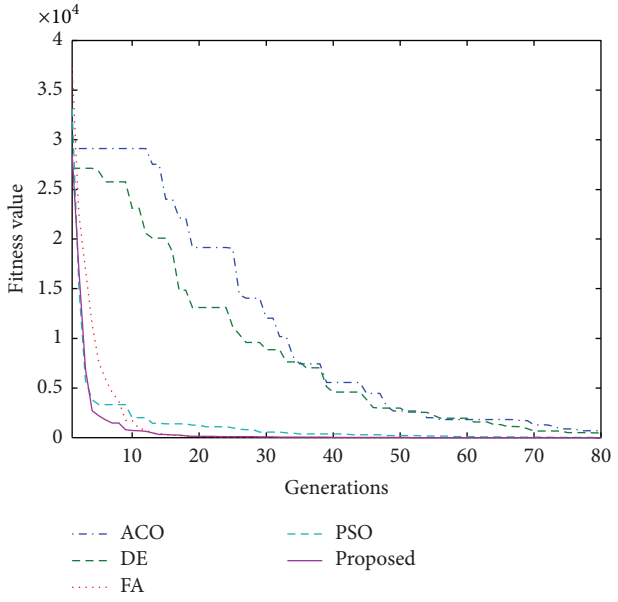

Figure 5: Maximum number of generations in which different algorithms minimize sphere function.

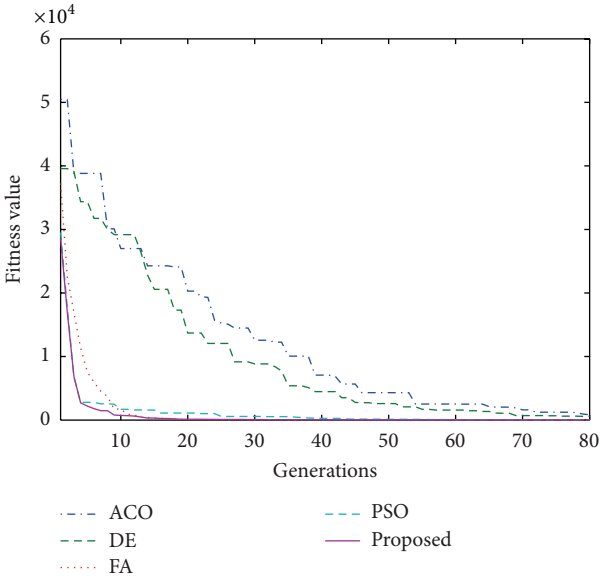

Figure 6: Shortest CPU times in terms of maximum number of generations for minimizing sphere function using different algorithms.

Table 4 illustrates the shortest and the longest CPU times based on ten simulation tests. Table 4 reveals that the proposed method yields a small fitness of $f_1(x)$ in the shortest CPU times and in the fewest generations. Figure 8 displays the number of generations of the different algorithmic methods and their shortest CPU times.

4.2. Benchmark Testing: Powell Function. The Powell $f_2(x) = \sum_{i=1}^{d/4}[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$, $-4 < x_i < 5$ where $d = 20$ and $x = (x_1, x_2, \ldots, x_{20})$ is a multimodal function whose global optimum value is $f_2(x) = 0$ and $x_1 = x_2 = x_3 = \cdots = x_{20} = 0$. Table 5 presents the best, mean, and worst values of minimizing Powell function obtained in 500 generations using all population-based algorithms. Figure 9 plots the performance of the algorithms in minimizing $f_2(x)$. Table 6 and Figure 10 display the shortest CPU times of the algorithm.

Table 5: Performance of methods used to minimize Powell function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$2.3302E-04$	$6.8694E-04$	$1.0520E-04$	$1.0321E-02$	$3.6326E-02$
	Time	1.7946	1.7312	11.7101	8.3304	1.6328
Mean	Fitness	$3.32096E-04$	$8.8461E-04$	$1.20681E-04$	$4.07067E-02$	$7.00505E-02$
	Time	1.7969	1.8037	11.6532	8.2524	1.6128
Worst	Fitness	$4.5921E-04$	$9.9689E-04$	$1.3828E-04$	$7.1160E-02$	$9.0721E-02$
	Time	1.8766	1.8609	11.7172	8.1815	1.5889

Table 6: Shortest CPU times for minimizing Powell function using various methods based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$4.0796E-04$	$9.0462E-04$	$1.1151E-04$	$2.6980E-02$	$5.3651E-02$
	Time	1.7684	1.7275	11.5836	8.1192	1.5722
Worst	Fitness	$4.5921E-04$	$9.9689E-04$	$1.3828E-04$	$3.6365E-02$	$5.0665E-02$
	Time	1.8766	1.8609	11.7172	8.3445	1.6441

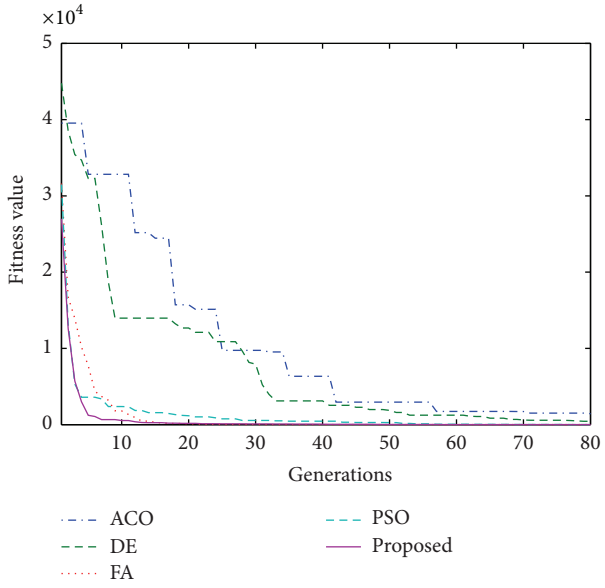

Figure 7: Convergence performance of different algorithms to minimize sphere function.

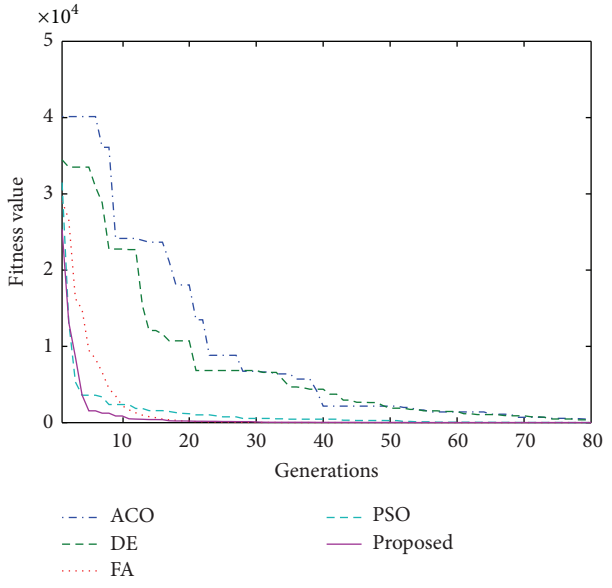

Figure 8: Shortest CPU times for convergence in minimizing sphere function using various algorithms.

Table 7 represents the best, mean, and worst values of the minimum $f_2(x)$ that is obtained using all techniques with the convergence tolerance set to 0.001. The proposed method yields the best value $f_2(x)$ in fewest generations based on ten simulation results. Figure 11 displays the minimum $f_2(x)$ at convergence. Table 8 provides the shortest and longest CPU times of the algorithms based on the ten simulation results. Table 8 indicates that the proposed method has shortest CPU times. Figure 12 displays the generation of the different population-based algorithms.

4.3. Benchmark Testing: Griewank Function. The Griewank function is given by $f_3(x) = \sum_{i=1}^{d}(x_i^2/4000) - \prod_{i=1}^{d}\cos(x_i/\sqrt{i}) + 1$, $-600 < x_i < 600$ where $d = 20$ and $x = (x_1, x_2, \ldots, x_{20})$. The Griewank function $f_3(x)$ is a highly multimodal function, whose global optimum value is $f_3(x) = 0$ and $x_1 = x_2 = x_3 = \cdots = x_{20} = 0$. Table 9 presents the best, mean, and worst values obtained using all of the tested algorithms in 500 generations. Figure 13 presents the performance of the different algorithms in minimizing $f_3(x)$. Table 10 and Figure 14 provide the shortest CPU times required by the various algorithms.

Table 11 presents the best, mean, and worst values that are used to minimize $f_3(x)$ using all techniques, based on ten simulation results. The convergence tolerance was set to 0.001. As presented, the proposed method provides the best mean value of $f_3(x)$ in the fewest generations and the shortest CPU time. Figure 15 shows the convergence

Table 7: Performance of methods used to minimize Powell function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$9.1273E-04$	$7.1296E-04$	$9.4488E-04$	$9.0360E-04$	$9.1736E-04$
	Generations	290	381	228	951	1941
	Time	0.9818	1.3197	5.3420	15.5685	6.1795
Mean	Fitness	$9.7111E-04$	$9.4425E-04$	$9.77216E-04$	$9.65246E-04$	$9.44765E-04$
	Generations	285.2	407.4	250.4	988.7	2084.1
	Time	1.0205	1.4655	5.8493	16.7481	6.6544
Worst	Fitness	$9.99680E-04$	$9.9668E-04$	$9.9727E-04$	$9.8909E-04$	$9.7555E-04$
	Generations	277	489	259	941	2040
	Time	1.0027	1.7057	5.9981	15.6953	6.3674

Table 8: Shortest CPU times of methods used to minimize Powell function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$9.9657E-04$	$8.8168E-04$	$9.9182E-04$	$9.8909E-04$	$9.7298E-04$
	Generations	265	348	228	767	1572
	Time	0.9722	1.2894	5.2914	13.1973	4.8964
Worst	Fitness	$9.8921E-04$	$9.9936E-04$	$9.8634E-04$	$9.5813E-04$	$9.3379E-04$
	Generations	308	489	281	1091	2385
	Time	1.1175	1.8205	6.5992	20.7070	7.6814

Table 9: Performance of various methods used to minimize Griewank function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$1.7037E-12$	$1.6977E-09$	$8.1102E-08$	$5.4901E-06$	$2.3019E-07$
	Time	1.5032	1.5415	8.7746	7.6647	1.3218
Mean	Fitness	$2.12356E-11$	$1.4780E-08$	$1.10972E-07$	$2.37129E-05$	$2.77944E-06$
	Time	1.4964	1.5460	8.9015	7.6075	1.3114
Worst	Fitness	$5.9058E-11$	$7.0892E-08$	$1.4187E-07$	$8.1614E-05$	$1.4428E-05$
	Time	1.5037	1.5000	8.9016	7.5366	1.3135

Table 10: Shortest CPU times in which various methods minimize Griewank function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$1.4997E-11$	$7.0892E-08$	$1.0660E-07$	$9.2675E-06$	$2.3775E-07$
	Time	1.4636	1.5000	8.7636	7.3618	1.2926
Worst	Fitness	$4.4415E-11$	$4.8823E-09$	$1.3706E-07$	$5.6119E-06$	$4.7007E-07$
	Time	1.5142	1.5975	9.0928	7.8357	1.3246

Table 11: Performance of different methods used to minimize Griewank function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$7.1941E-04$	$8.6557E-04$	$7.1756E-04$	$8.0627E-04$	$6.9771E-04$
	Generations	203	182	280	458	356
	Time	0.5851	0.5725	5.0284	6.9718	0.9431
Mean	Fitness	$9.0097E-04$	$9.5185E-04$	$8.98598E-04$	$9.09750E-04$	$9.00905E-04$
	Generations	193.9	195.6	278.4	415.6	378.8
	Time	0.5788	0.6029	4.9622	6.3172	0.9962
Worst	Fitness	$9.91600E-04$	$9.9607E-04$	$9.8911E-04$	$9.9483E-04$	$9.9493E-04$
	Generations	192	186	283	418	413
	Time	0.5737	0.5704	5.0228	6.4388	1.0912

Table 12: Shortest CPU times of different methods used to minimize Griewank function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$9.0505E-04$	$9.9607E-04$	$9.4839E-04$	$9.0539E-04$	$9.3450E-04$
	Generations	189	186	272	399	354
	Time	0.5494	0.5704	4.8042	5.8471	0.9111
Worst	Fitness	$8.5759E-04$	$9.9382E-04$	$9.1605E-04$	$8.0627E-04$	$9.9493E-04$
	Generations	204	232	283	458	413
	Time	0.6176	0.7174	5.2167	6.9718	1.0912

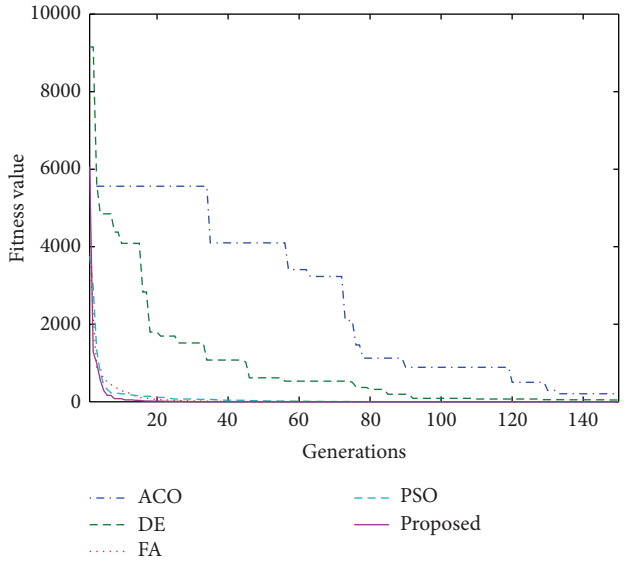

Figure 9: Maximum number of generations in which algorithms minimize Powell function.

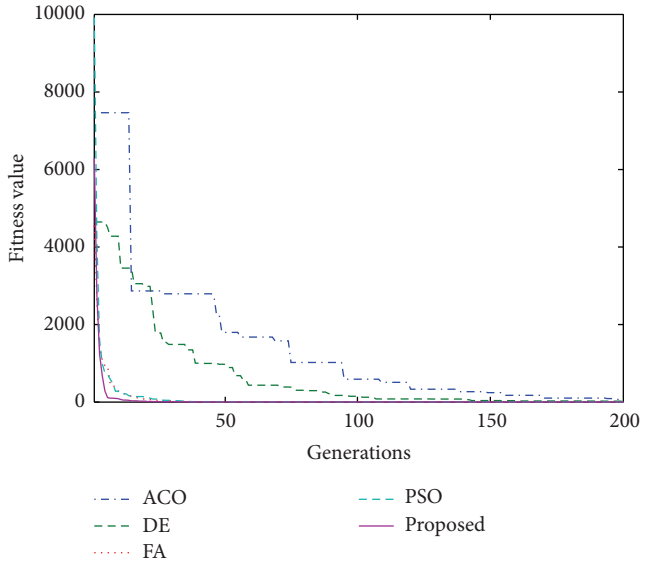

Figure 10: Shortest CPU times in terms of maximum number of generations in which algorithms minimize Powell function.

performance of each method in minimizing $f_3(x)$. Table 12 presents the shortest and the longest CPU times based on the ten simulation tests of the different algorithms. Table 12 shows that the proposed method yields the smallest value of $f_3(x)$ in the shortest CPU times. Figure 16 plots the generation of the different algorithms.

4.4. Benchmark Testing: Ackley Function. The Ackley function $f_4(x) = -20e^{-0.2\times\sqrt{(1/d)\sum_{i=1}^{d} x_i^2}} - e^{(1/d)\sum_{i=1}^{d}\cos(2\pi\times x_i)} + 20 + e^1$, $-32 < x_i < 32$ where $d = 20$ and $x = (x_1, x_2, \ldots, x_{20})$ is a multimodal function whose global optimum value is $f_4(x) = 0$ and $x_1 = x_2 = x_3 = \cdots = x_{20} = 0$. Table 13 presents the best, mean, and worst values obtained using all of the algorithms in 500 generations. Figure 17 presents the performance of the algorithms in minimizing $f_4(x)$. Table 14 and Figure 18 highlight the shortest CPU times of the different population-based algorithms.

Table 15 presents the best, mean, and worst values of the minimum $f_4(x)$ that are obtained using all techniques when the convergence tolerance was set to 0.001. Based on the ten simulation results, the proposed method provides best optimal value of $f_4(x)$ in the fewest generations. Figure 19 plots the convergence value in the minimizing of $f_4(x)$. Table 16 presents the shortest and longest CPU times required by the different algorithms based on the ten simulation results. Table 16 reveals that the proposed method yields a smallest fitness value of $f_4(x)$. Figure 20 plots the generation of the different population-based algorithms.

4.5. FLC Optimized by Chaos-Enhanced PSO with Adaptive Parameters for Maximum Power Point Tracking in Standalone Photovoltaic System. Developing fuzzy logic control for the MPPT [61–67] involves determining the scaling factor parameters and the shape of the fuzzy membership functions. The two inputs and one output for this purpose are $E(n)$, which is tracking error, $\Delta E(n)$, which is change of tracking error, and $\Delta D(n)$, which is the change of the duty cycle. They are selected to tune optimally the fuzzy logic controller. The mathematical description are given as $E(n) = (p(n) - p(n-1))/(v(n) - v(n-1))$ and $\Delta E(n) = E(n) - E(n-1)$. In this case, $p(n)$ and $v(n)$ are the instantaneous power and voltage of the PV, respectively. $E(n)$ represents the operating power point of the load, whether it is currently located on the left hand side or right hand side, while $\Delta E(n)$ denotes

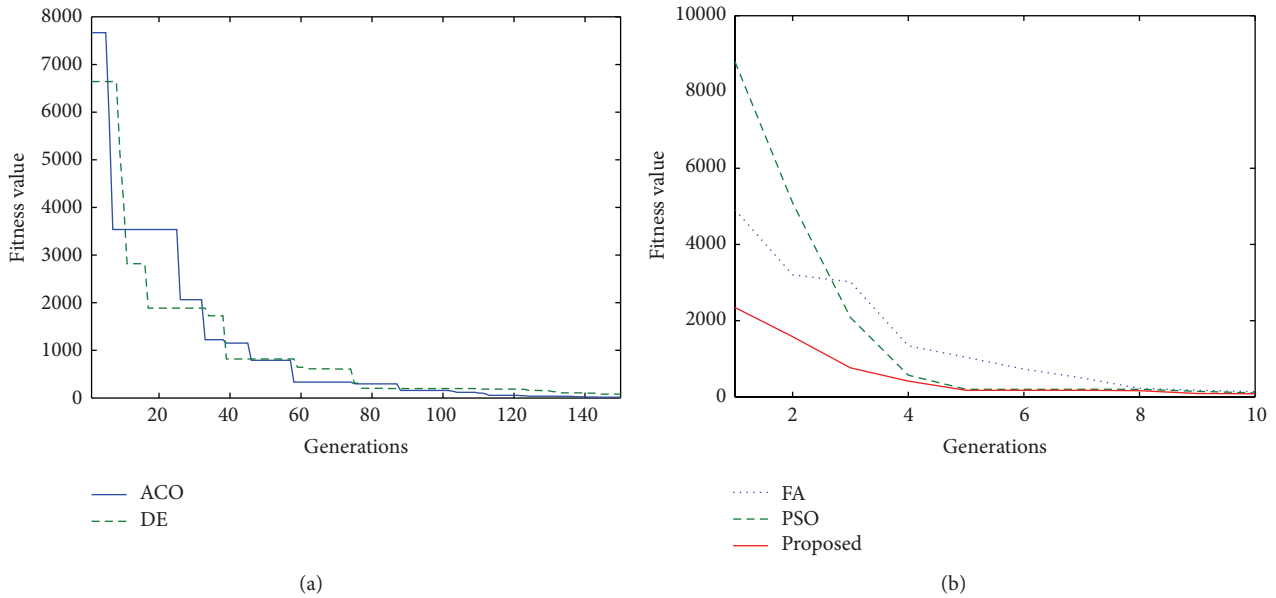

Figure 11: Convergence performance of algorithms used to minimize Powell function: (a) ACO and DE; (b) FA, PSO, and proposed.

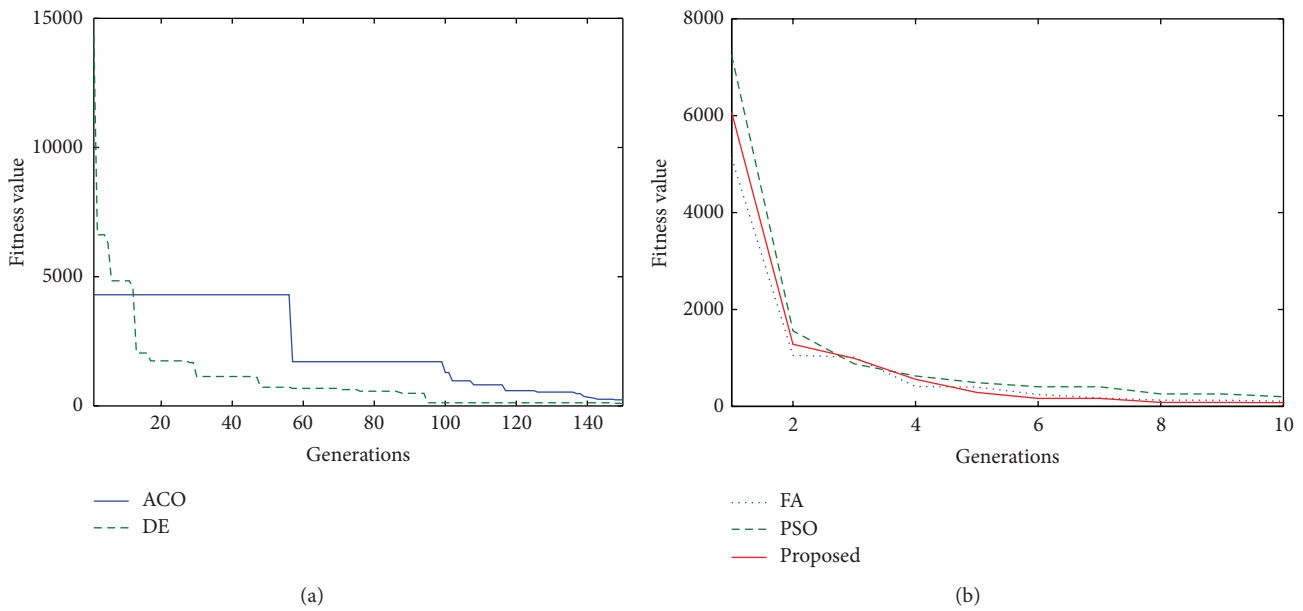

Figure 12: Shortest CPU times performance for convergence of algorithms used to minimize Powell function: (a) ACO and DE; (b) FA, PSO, and proposed.

Table 13: Performance of different methods used in minimizing Ackley function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$1.7320E-07$	$1.6304E-05$	$5.0558E-05$	$4.9309E-05$	$2.8386E-05$
	Time	1.4763	1.6495	9.5650	7.6734	1.4923
Mean	Fitness	$4.44804E-07$	$3.1875E-05$	$5.92199E-05$	$7.77767E-05$	$5.79878E-05$
	Time	1.4848	1.6477	9.5221	7.6975	1.4897
Worst	Fitness	$7.1067E-07$	$9.3937E-05$	$6.5692E-05$	$9.6797E-05$	$8.8300E-05$
	Time	1.5081	1.6460	9.5146	7.7872	1.4923

Table 14: Shortest CPU times of different methods in minimizing Ackley function based on ten simulation results (number of generations = 500).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$2.1591E-07$	$2.3595E-05$	$5.5490E-05$	$7.6737E-05$	$5.1565E-05$
	Time	1.4625	1.5487	9.4085	7.5984	1.4536
Worst	Fitness	$3.6531E-07$	$4.2848E-05$	$6.2945E-05$	$8.5803E-05$	$7.7403E-05$
	Time	1.4996	1.7014	9.7459	7.9017	1.6048

Table 15: Performance of different methods used to minimize Ackley function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$8.9400E-04$	$2.5891E-04$	$8.6650E-04$	$8.5899E-04$	$8.5410E-04$
	Generations	230	278	358	449	407
	Time	0.6973	0.9269	6.6928	7.0274	1.2136
Mean	Fitness	$9.41E-04$	$8.2176E-01$	$9.41769E-04$	$7.79853E-04$	$9.37661E-04$
	Generations	245.8	264.3	362.3	433.1	397.7
	Time	0.7433	0.8218	6.8563	4.9292	1.1824
Worst	Fitness	$9.93800E-04$	$9.9982E-04$	$9.9644E-04$	$9.9878E-04$	$9.9868E-04$
	Generations	248	237	362	353	417
	Time	0.7412	0.7587	6.7990	6.7089	1.2317

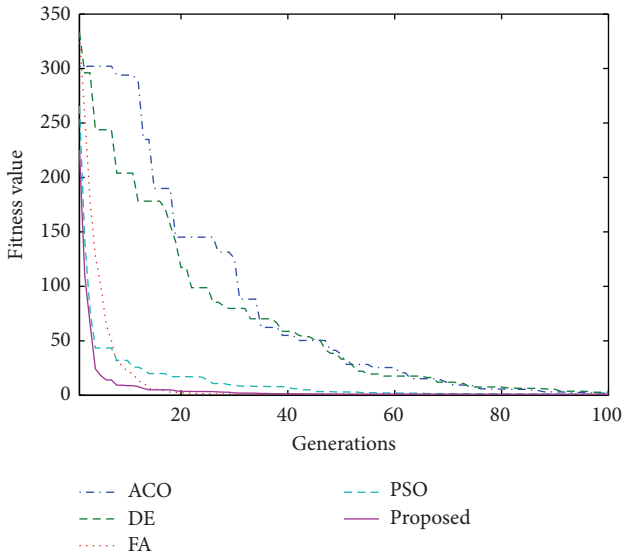

Figure 13: Maximum number of generations in which algorithms minimize Griewank function.

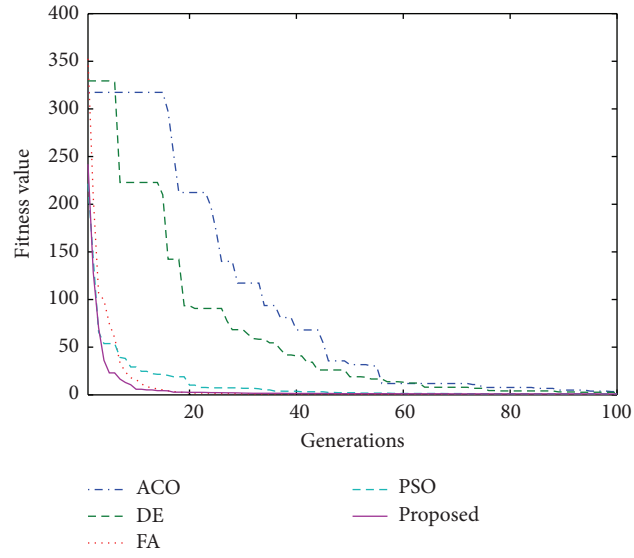

Figure 14: Shortest CPU times in terms of maximum number of generations in which various algorithms minimize Griewank function.

the direction of motion of the operating point. The fuzzy inference system (FIS) approach used herein for maximum power point tracking was the Mamdani system with the min-max fuzzy combination operation for maximum power point tracking. The defuzzification method that was used to obtain the actual value of the duty cycle signal as a crisp output was the center of gravity-based method. The equation is $D = \sum_{i=1}^{n}(\mu(D_i) - D_i)/\sum_{i=1}^{n}\mu(D_i)$. The variable output is the pulse width modulation signal, which is transmitted to the DC/DC boost converter to drive the necessary load (Table 18). The chaos-enhanced particle swarm optimization with adaptive parameters was utilized to determine the parameter of the scaling factors and to optimize the width of each inputs and output membership function. Each of these inputs and outputs includes five membership functions of the fuzzy logic.

The operation of the chaos-enhanced PSO with adaptive parameters begins by generating a solution from the randomly generated population with the best positions. The velocity equation yields particles in better positions through the application of chaos search and the self-organizing

Table 16: Shortest CPU times of different methods used to minimize Ackley function based on ten simulation results (convergence tolerance = 0.001).

		Proposed	PSO	FA	ACO	DE
Best	Fitness	$8.9400E-04$	$9.7640E-04$	$8.6650E-04$	$9.3713E-04$	$9.9383E-04$
	Generations	230	233	358	385	369
	Time	0.6973	0.7319	6.6928	6.0902	1.0919
Worst	Fitness	$9.7772E-04$	$2.5891E-04$	$8.8023E-04$	$9.7779E-04$	$9.9868E-04$
	Generations	256	278	368	465	417
	Time	0.7770	0.9269	7.0483	7.3161	1.2317

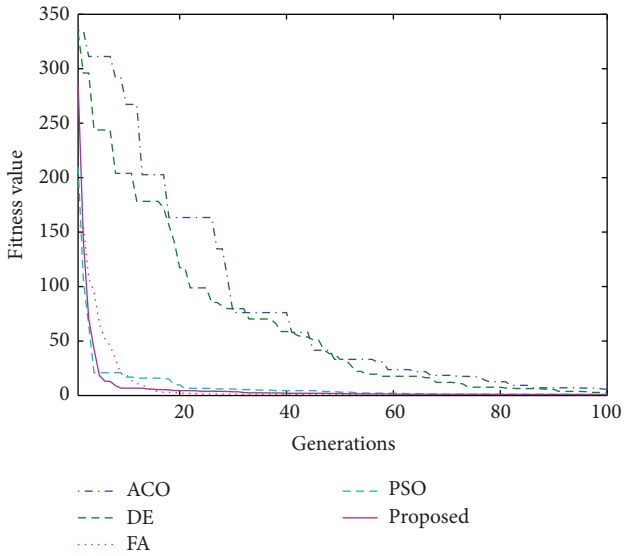

Figure 15: Convergence performance of different algorithms used to minimize Griewank function.

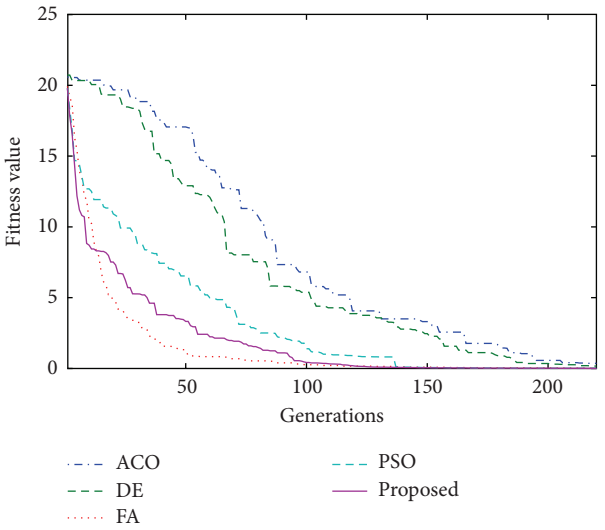

Figure 17: Maximum number of generations in which different algorithms minimize Ackley function.

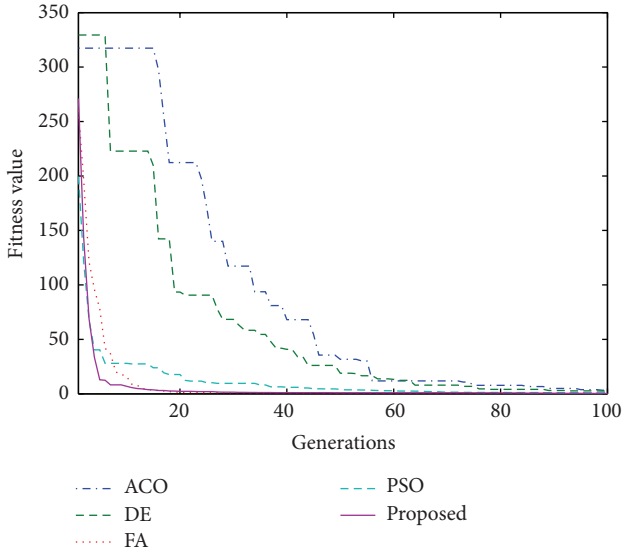

Figure 16: Shortest CPU times for convergence of different algorithms to minimize Griewank function.

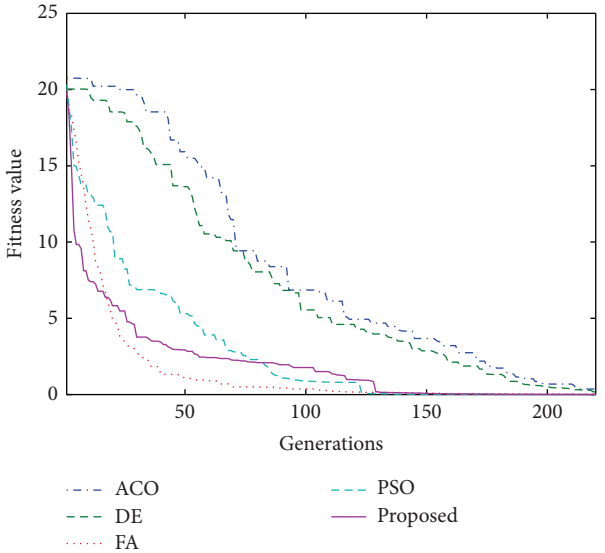

Figure 18: Shortest CPU times in terms of maximum number of generations in which different algorithms minimize Ackley function.

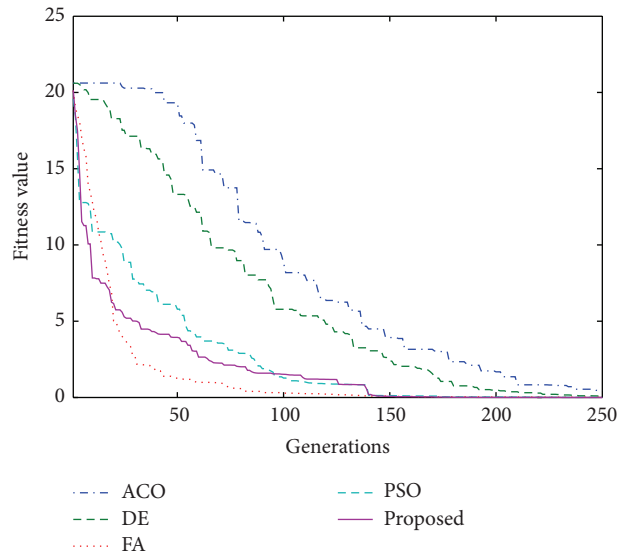

Figure 19: Convergence performance of different algorithms used to minimize Ackley function.

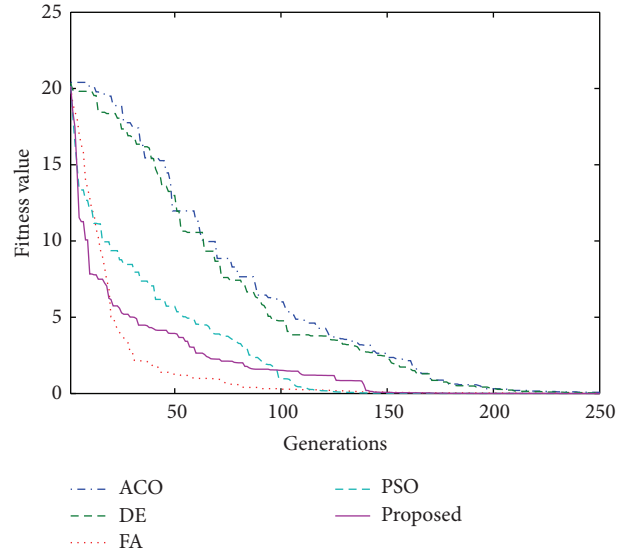

Figure 20: Shortest CPU times in terms of convergence for different algorithms used to minimize Ackley function.

parameters. In this paper, the cost function that is used as the performance index is based on the minimization of the integral absolute error (IAE): ObjFunc $= \int_0^\infty |p(t)_{\text{ref}} - p(t)_{\text{out}}|dt$. The fitness value is calculated using Fitness $=$ 2000 $-$ ObjFunc. The measured cost function yields the dynamic maximum output power of the boost converter. The maximum power point tracking control was carried out using the fuzzy logic and scaling factor controllers. Figure 21 presents the model that was used to tune the parameters of the fuzzy logic controller during the process of optimization. The benchmark test is conducted with a variable irradiance and temperature of PV module as operating conditions, which is shown in Figure 22 and the DC/DC boost converter

Table 17: SunPower SPR-305-WHT PV array electrical characteristics.

Parameters	Variable	Value
Total number of cells	N_{cell}	96
Number of series connected modules per string	N_{ser}	1
Number of parallel strings	N_{par}	4
Open circuit voltage	V_{oc}	64.2 V
Short circuit current	I_{sc}	5.96 A
Voltage at maximum power	V_m	54.7 V
Current at maximum power	I_m	5.58 A
Maximum power (±5%)	P_m	305 W
Maximum system voltage	IEC, UL	1000 V, 600 V
Reference cell temperature	T_{ref}	25°C
Reference irradiation	S_{ref}	1000 W/m^2

Table 18: DC/DC boost converter specifications.

Parameters	Variable	Value
Input filter capacitance	C_{if}	2 mF
Boost inductance	L	0.01 H
Output filter capacitance	C_{of}	2 mF
Load resistance	R_L	500 Ω

Table 19: Fuzzy logic control rule base.

		Change in error $\Delta E(n)$				
		NB	*NS*	*ZE*	*PS*	*PB*
Error $E(n)$	*NB*	ZE	ZE	PB	PB	PB
	NS	ZE	ZE	PS	PS	PS
	ZE	PS	ZE	ZE	ZE	NS
	PS	NS	NS	NS	ZE	ZE
	PB	NB	NB	NB	ZE	ZE

is utilized to validate the optimized fuzzy logic maximum power point tracking controller. All updates and transfer of data are executed as set in the model. During the generation process, the parameters of the fuzzy logic controllers and the scaling factors are updated. These parameters are retained until a new global fitness is obtained during the optimization process. At the end of each generation, the parameters in the fuzzy logic and the scaling factors are updated based on the obtained global fitness until a convergence is made for the best solution found so far by the swarm. The Appendix presents the solar PV array specifications (also see Table 17), the parameters used for DC/DC boost converter [68–72], and the rule base of the fuzzy logic controller (Table 19). Figure 23 displays the optimal best inputs and output width of membership functions obtained by the swarm. The optimal fuzzy logic solution yields symmetric triangular membership functions for $E(n)$, $\Delta E(n)$, and $\Delta D(n)$. The chaos-enhanced PSO with adaptive parameters causes the maximum power point tracking of a PV system to converge toward the best fitness that has been obtained by the swarm. Figure 24 shows

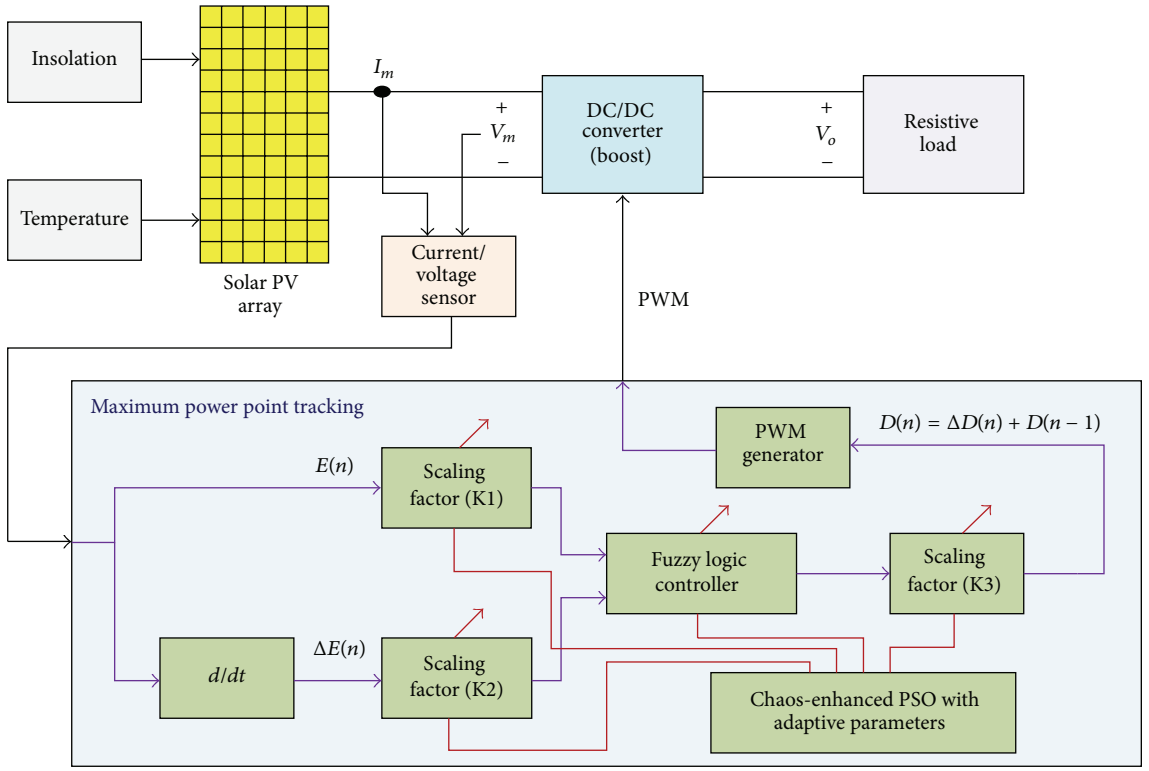

Figure 21: Block diagram of maximum power point tracking for standalone PV system.

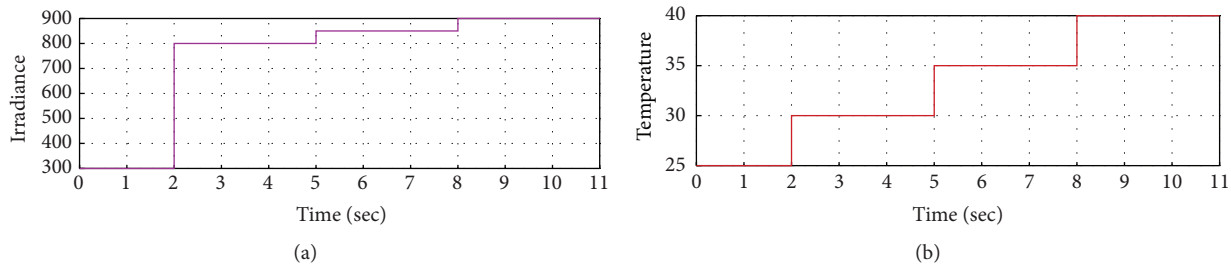

Figure 22: Staircase change of solar irradiation (300 to 900 W/m^2) and temperature (25°C to 40°C).

the output power. The obtained optimal fuzzy logic controller is more robust than, and outperforms, other maximum power point tracking algorithms.

5. Conclusion

This paper presents a novel technique with promising new features to enhance the performance and robustness of the standard PSO for solving optimization problems. The improved technique incorporates chaos searching to avoid the risk of stagnation, premature convergence, and trapping in a local optimum; it incorporates type $1''$ constriction to improve the quality of convergence and adaptive cognitive and social learning coefficients to improve the exploitation and exploration search characteristics of the algorithm. The proposed chaos-enhanced PSO with adaptive parameters was experimentally tested in high-dimensional problem space using a four benchmark functions to verify its effectiveness. The advantages of the chaos-enhanced PSO with adaptive parameters over the population-based algorithms are verified and the numerical results demonstrate that the proposed technique offers a faster convergence with near precise results, better reliability, and lower computational burden; avoids stagnation and premature convergence; and can escape from local minimum and low CPU time and speed requirements. A complete stand-alone PV model was developed in which maximum power point tracking control with fuzzy logic is utilized to evaluate the performance of the proposed algorithm in real-world engineering optimization applications.

It is envisaged that the proposed chaos-enhanced PSO with adaptive parameters can be applied to a wider class of complex scientific and engineering problems such as electric power system optimization (e.g., minimization of nonconvex fuel cost and power losses), robust design

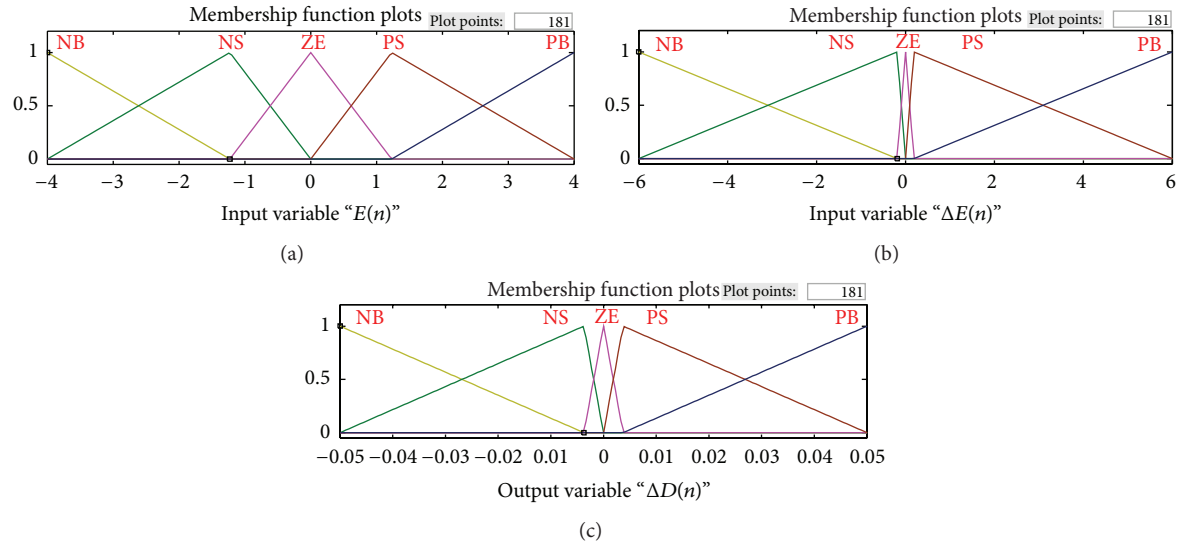

FIGURE 23: Graphical representation of obtained optimal fuzzy logic membership function for inputs (a) $E(n)$ and (b) $\Delta E(n)$ and output (c) $\Delta D(n)$ linguistic variables.

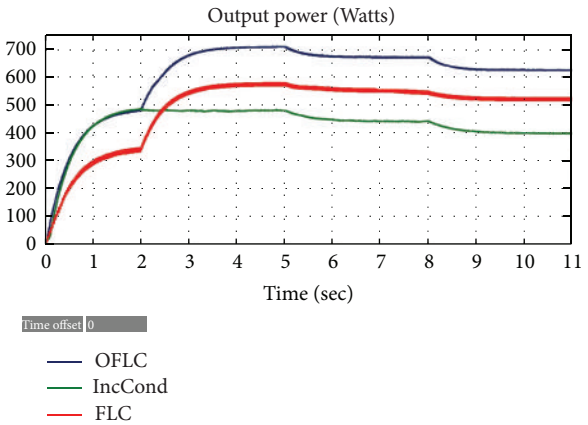

FIGURE 24: Output power response of optimal tuned fuzzy logic (OFLC), incremental conductance (IncCond), and fuzzy logic (FLC) under variable irradiance and temperature.

of nonlinear plant control system under the presence of parametric uncertainties, and forecasting of wind farm output power using the artificial neural network where the connection weights and thresholds are needed to be adjusted optimally.

Appendix

This appendix presents the solar PV array specifications [73], DC/DC boost converter parameters, and fuzzy logic rule base of the five membership functions that are used in the DC/DC boost converter.

The maximum power for a single PV array (watts) is given as follows:

$$P_{\text{array}} = P_m \times N_{\text{ser}} \times N_{\text{par}}. \quad \text{(A.1)}$$

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the Ministry of Science and Technology of the Republic of China, Taiwan, for financially supporting this research under Contract MOST 104-2221-E-033-029.

References

[1] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Wash, USA, December 1995.

[3] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015.

[4] S. Bouallègue, J. Haggège, and M. Benrejeb, "Particle swarm optimization-based fixed-structure $\mathscr{H}_\infty$ control design," *International Journal of Control, Automation and Systems*, vol. 9, no. 2, pp. 258–266, 2011.

[5] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 81–86, Seoul, Republic of Korea, May 2001.

[6] F. Van den Bergh, *An analysis of particle swarm optimizers [Ph.D. thesis]*, University of Pretoria, Pretoria, South Africa, 2006.

[7] E. P. Ruben and B. Kamran, "Particle swarm optimization in structural design," in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, F. T. S. Chan and M. K. Tiwari, Eds., pp. 373–394, I-Tech Education and Publication, Vienna, Austria, 2007.

[8] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*, IGI Global, Hershey, Pa, USA, 2010.

[9] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[10] H.-Q. Li and L. Li, "A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems," in *Proceedings of the International Conference on Intelligent Pervasive Computing (IPC '07)*, pp. 94–97, IEEE, Jeju, South Korea, October 2007.

[11] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in solar photovoltaic system," *Applied Soft Computing Journal*, vol. 13, no. 5, pp. 2997–3006, 2013.

[12] B. Samanta and C. Nataraj, "Use of particle swarm optimization for machinery fault detection," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 2, pp. 308–316, 2009.

[13] L. Liu, W. Liu, and D. A. Cartes, "Particle swarm optimization-based parameter identification applied to permanent magnet synchronous motors," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 7, pp. 1092–1100, 2008.

[14] H. Modares, A. Alfi, and M.-M. Fateh, "Parameter identification of chaotic dynamic systems through an improved particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3714–3720, 2010.

[15] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[16] W. Jiekang, Z. Jianquan, C. Guotong, and Z. Hongliang, "A hybrid method for optimal scheduling of short-term electric power generation of cascaded hydroelectric plants based on particle swarm optimization and chance-constrained programming," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1570–1579, 2008.

[17] Y.-Y. Hong, F.-J. Lin, Y.-C. Lin, and F.-Y. Hsu, "Chaotic PSO-based VAR control considering renewables using fast probabilistic power flow," *IEEE Transactions on Power Delivery*, vol. 29, no. 4, pp. 1666–1674, 2014.

[18] Y. Marinakis, M. Marinaki, and G. Dounias, "A hybrid particle swarm optimization algorithm for the vehicle routing problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 463–472, 2010.

[19] G. K. Venayagamoorthy, S. C. Smith, and G. Singhal, "Particle swarm-based optimal partitioning algorithm for combinational CMOS circuits," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 2, pp. 177–184, 2007.

[20] S. Bouallègue, J. Haggège, M. Ayadi, and M. Benrejeb, "PID-type fuzzy logic controller tuning based on particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 484–493, 2012.

[21] Y.-Y. Hong, F.-J. Lin, S.-Y. Chen, Y.-C. Lin, and F.-Y. Hsu, "A Novel adaptive elite-based particle swarm optimization applied to VAR optimization in electric power systems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 761403, 14 pages, 2014.

[22] S. Saini, D. R. B. A. Rambli, M. N. B. Zakaria, and S. B. Sulaiman, "A review on particle swarm optimization algorithm and its variants to human motion tracking," *Mathematical Problems in Engineering*, vol. 2014, Article ID 704861, 16 pages, 2014.

[23] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[24] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[25] H. Wang, C. Li, Y. Liu, and S. Zeng, "A hybrid particle swarm algorithm with cauchy mutation," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 356–360, IEEE, Honolulu, Hawaii, USA, April 2007.

[26] H. Wang, Y. Liu, S. Zeng, H. Li, and C. Li, "Opposition-based particle swarm algorithm with cauchy mutation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 4750–4756, IEEE, Singapore, September 2007.

[27] M. Pant, T. Radha, and V. P. Singh, "Particle swarm optimization using gaussian inertia weight," in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, vol. 1, pp. 97–102, Sivakasi, India, December 2007.

[28] T. Xiang, K.-W. Wong, and X. Liao, "A novel particle swarm optimizer with time-delay," *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 789–793, 2007.

[29] Z. Cui, X. Cai, J. Zeng, and G. Sun, "Particle swarm optimization with FUSS and RWS for high dimensional functions," *Applied Mathematics and Computation*, vol. 205, no. 1, pp. 98–108, 2008.

[30] M. A. Montes de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.

[31] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[32] R. Eberhart and Y. Shi, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 591–600, San Diego, Calif, USA, March 1998.

[33] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, vol. 1, pp. 84–88, La Jolla, Calif, USA, July 2000.

[34] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 6, pp. 1272–1282, 2005.

[35] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[36] Y.-T. Juang, S.-L. Tung, and H.-C. Chiu, "Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions," *Information Sciences*, vol. 181, no. 20, pp. 4539–4549, 2011.

[37] P. S. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for

improved continuous optimization," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 129–142, 2007.

[38] B. Xin, J. Chen, J. Zhang, H. Fang, and Z.-H. Peng, "Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 5, pp. 744–767, 2012.

[39] A. Kaveh and V. R. Mahdavi, "A hybrid CBO-PSO algorithm for optimal design of truss structures with dynamic constraints," *Applied Soft Computing*, vol. 34, pp. 260–273, 2015.

[40] M. S. Innocente and J. Sienz, "Particle swarm optimization with inertia weight and constriction factor," in *Proceedings of the International Joint Conference on Swarm Intelligence (ICSI '11)*, pp. 1–11, EISTI, Cergy, France, June 2011.

[41] Y.-H. Liu, S.-C. Huang, J.-W. Huang, and W.-C. Liang, "A particle swarm optimization-based maximum power point tracking algorithm for PV systems operating under partially shaded conditions," *IEEE Transactions on Energy Conversion*, vol. 27, no. 4, pp. 1027–1035, 2012.

[42] E. Ott, C. Grebogi, and J. A. Yorke, "Controlling chaos," *Physical Review Letters*, vol. 64, no. 11, pp. 1196–1199, 1990.

[43] L. Liu, Z. Han, and Z. Fu, "Non-fragile sliding mode control of uncertain chaotic systems," *Journal of Control Science and Engineering*, vol. 2011, Article ID 859159, 6 pages, 2011.

[44] E. N. Lorenz, "Deterministic non periodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 11, pp. 131–141, 1963.

[45] V. Pediroda, L. Parussini, C. Poloni, S. Parashar, N. Fateh, and M. Poian, "Efficient stochastic optimization using chaos collocation method with modefrontier," *SAE International Journal of Materials and Manufacturing*, vol. 1, no. 1, pp. 747–753, 2009.

[46] Y. Huang, P. Zhang, and W. Zhao, "Novel grid multiwing butterfly chaotic attractors and their circuit design," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 5, pp. 496–500, 2015.

[47] X. H. Mai, D. Q. Wei, B. Zhang, and X. S. Luo, "Controlling chaos in complex motor networks by environment," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 6, pp. 603–607, 2015.

[48] Z. Wang, J. Chen, M. Cheng, and K. T. Chau, "Field-oriented control and direct torque control for paralleled VSIs Fed PMSM drives with variable switching frequencies," *IEEE Transactions on Power Electronics*, vol. 31, no. 3, pp. 2417–2428, 2016.

[49] J. D. Morcillo, D. Burbano, and F. Angulo, "Adaptive ramp technique for controlling chaos and subharmonic oscillations in DC-DC power converters," *IEEE Transactions on Power Electronics*, vol. 31, no. 7, pp. 5330–5343, 2016.

[50] G. Kaddoum and F. Shokraneh, "Analog network coding for multi-user multi-carrier differential chaos shift keying communication system," *IEEE Transactions on Wireless Communications*, vol. 14, no. 3, pp. 1492–1505, 2015.

[51] J. M. Valenzuela, "Adaptive anti control of chaos for robot manipulators with experimental evaluations," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 1, pp. 1–11, 2013.

[52] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, p. 475, Kumamoto, Japan, September 2007.

[53] M. Ausloos and M. Dirickx, *The Logistic Map and the Route to Chaos*, Understanding Complex Systems, Springer, Berlin, Germany, 2006.

[54] A. M. Arasomwan and A. O. Adewumi, "An investigation into the performance of particle swarm optimization with various chaotic maps," *Mathematical Problems in Engineering*, vol. 2014, Article ID 178959, 17 pages, 2014.

[55] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, October 26–28, 2009. Proceedings*, vol. 5792 of *Lecture Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[56] X. S. Yang, *Engineering Optimisation: An Introduction with Metaheuristic Applications*, John Wiley and Sons, New York, NY, USA, 2010.

[57] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[58] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[59] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS '96)*, pp. 519–523, Berkeley, Calif, USA, June 1996.

[60] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[61] J.-K. Shiau, M.-Y. Lee, Y.-C. Wei, and B.-C. Chen, "Circuit simulation for solar power maximum power point tracking with different buck-boost converter topologies," *Energies*, vol. 7, no. 8, pp. 5027–5046, 2014.

[62] J.-K. Shiau, Y.-C. Wei, and B.-C. Chen, "A study on the fuzzy-logic-based solar power MPPT algorithms using different fuzzy input variables," *Algorithms*, vol. 8, no. 2, pp. 100–127, 2015.

[63] P.-C. Cheng, B.-R. Peng, Y.-H. Liu, Y.-S. Cheng, and J.-W. Huang, "Optimization of a fuzzy-logic-control-based MPPT algorithm using the particle swarm optimization technique," *Energies*, vol. 8, no. 6, pp. 5338–5360, 2015.

[64] A. Mellit, A. Messai, A. Guessoum, and S. A. Kalogirou, "Maximum power point tracking using a GA optimized fuzzy logic controller and its FPGA implementation," *Solar Energy*, vol. 85, no. 2, pp. 265–277, 2011.

[65] C. Larbes, S. M. Aït Cheikh, T. Obeidi, and A. Zerguerras, "Genetic algorithms optimized fuzzy logic control for the maximum power point tracking in photovoltaic system," *Renewable Energy*, vol. 34, no. 10, pp. 2093–2100, 2009.

[66] L. K. Letting, J. L. Munda, and Y. Hamam, "Optimization of a fuzzy logic controller for PV grid inverter control using S-function based PSO," *Solar Energy*, vol. 86, no. 6, pp. 1689–1700, 2012.

[67] R. Ramaprabha, M. Balaji, and B. L. Mathur, "Maximum power point tracking of partially shaded solar PV system using modified Fibonacci search method with fuzzy controller," *International Journal of Electrical Power and Energy Systems*, vol. 43, no. 1, pp. 754–765, 2012.

[68] K. C. Wu, *Pulse Width Modulated DC-DC Converters*, Springer, New York, NY, USA, 1997.

[69] F. L. Luo and H. Ye, *Advanced DC/DC Converters*, CRC Press, 2003.

[70] H. Sira-Ramirez and R. Silva-Ortigoza, *Control Design Techniques in Power Electronics Devices*, Springer, London, UK, 2006.

[71] M. K. Kazimierczuk, *Pulse-Width Modulated DC-DC Power Converters*, John Wiley & Sons, 2008.

[72] M. H. Rashid, *Electric Renewable Energy Systems*, Academic Press, Cambridge, Mass, USA, 2016.

[73] SunPower Corporation, *SunPower 305 Solar Panel*, SunPower Corporation, San Jose, Calif, USA, 2009.