PV PARAMETERS ESTIMATION USING DIFFERENT EVOLUTIONARY ALGORITHMS

Mohamed AZAB
Faculty of Engineering
Benha University
Egypt
Pe_bhit@yahoo.com

Fawzan SALEM
Electronics Research Institute,
Egypt
fawzan@lycos.com

M. I. MOSAAD
Department of Electrical Engineering
Higher Technological Institute HTI
Egypt
m_i_mosaad@hotmail.com

Abstract: *In this paper, different evolutionary algorithms have been employed to estimate the parameters of a real PV module. The used algorithms are Particle Swarm Bacterial Foraging, Simulated Annealing and Genetic Algorithms. The results endorse the capability of evolutionary computational approaches to determine precisely the parameters of the PV module. However, some differences have been observed in terms of required number of iterations and accuracy. The Particle Swarm and Bacterial Foraging showed noteworthy precision compared to Genetic Algorithm and Simulated Annealing in handling this problem. According to the carried out studies, Evolutionary Algorithms provided several sets of solutions to the estimation problem rather than a unique solution.*

Keywords: *PV parameters estimation, particle swarm optimization, bacterial foraging optimization, simulated annealing optimization, genetic algorithms.*

1. Introduction

In the last few decades, PV systems became common in grid-connected applications. However, unlike traditional power plants, cost and performance of PV systems strongly depend on the electrical properties (parameters) of the modules. Therefore, PV system investment decisions and PV system designing are not easy tasks. To overcome this problem, several analytical methods [1-4] and evolutionary computational algorithms [5-8] were developed to estimate the PV system parameters.

Authors in [1] have demonstrated a step-by-step methodology based on Gauss–Seidel numerical method to estimate the lumped equivalent parameters of a PV module from its datasheets provided by manufacturers. Another analytical method is presented in [2] to estimate the most important parameters of a PV module using manufacturer's datasheet. Authors in [3, 4] have introduced two analytical methods based on Monte Carlo simulations and it was found that the estimated parameters were in agreement with the theoretical expression of the uncertainty.

Authors in [5] have presented an approach for improving the extracting accuracy of the PV module parameters using a combination of an adaptive Genetic Algorithm (GA) with a Least Squares Gradient search. In [6], a new parameter extraction method based on the Differential Evolution (DE) technique was introduced. The performance of DE is evaluated against GA using a synthetic and experimental I-V data set. It is found that the DE method fits the I–V curve better than GA, has a lower fitness function value and faster execution time. Authors in [7, 8] have demonstrated Swarm Intelligence approach to extract equivalent circuit parameters of PV modules. It has been confirmed that the two approaches can obtain good parameter precision under the variations of solar radiation and environmental temperature.

In this paper, four different evolutionary algorithms are introduced to estimate the parameters of a PV module and comparisons between each of them. The algorithms are Particle Swarm Optimization (PSO), Bacterial Foraging Optimization (BFO), Simulated Annealing Optimization (SAO) and Genetic Algorithm (GA).

This paper is organized as follows: In section 2 the mathematical model of a PV module is presented. Problem formulation is covered in section 3. Section 4 is assigned to present the four evolutionary

computational algorithms used for parameter estimation including their steps. Section 5 demonstrates and compares between the results obtained using these four algorithms with the experimental results. Finally, section 6 concludes the entire paper.

2. PV Modeling

The commonly used equivalent circuit of a PV solar cell is shown in Fig. 1, where the model consists of a light dependent current source in parallel with an equivalent diode structure. The output of the current source is directly proportional to the light falling on the cell. The solar cell fails to maintain a fixed current as the load resistance increases. The output current reaches to zero when the load resistance becomes very large.

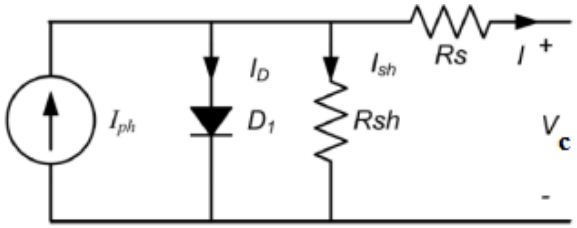

Fig. (1). Solar cell equivalent circuit

The solar cell current is determined as [9]:

$$I_c = I_{ph} - I_o\left\{e^{\left[\frac{q}{AKT}(V_c + I_c R_s)\right]} - 1\right\} - \frac{V_c + I_c R_s}{R_{sh}} \qquad (1)$$

Where I_c is the cell current, I_{ph} is the light generated current, I_o is the reverse saturation current, q is the electron charge ($1.6\text{x}10^{-19}$C), A is the ideality factor, K is the Boltzman constant ($1.38\text{x}10^{-23}$ Nm/^{0}K), T is the cell temperature (^{o}K), V_c is the cell voltage, R_s is the series resistance, and R_{sh} is the shunt resistance.

The photoelectric current and reverse saturation current of the solar cell can be calculated, respectively, using the following formulas:

$$I_{ph} = \frac{G}{1000}\left[I_{sc} + k_i\left(T - T_r\right)\right] \qquad (2)$$

$$I_o = I_{or}\left(\frac{T}{T_r}\right)^3 e^{\left[\frac{qE_g}{AK}\left(\frac{1}{T_r} - \frac{1}{T}\right)\right]} \qquad (3)$$

Where G is the radiation (w/m^2), I_{sc} is the short circuit current, K_i is the short circuit current temperature coefficient ($A/^oC$), T_r is the reference temperature (298K), I_o is reverse saturation current, I_{or} is reverse saturation current at T_r, and E_g is band gap for silicon (1.1 ev).

The PV module consists of series connected solar cells (n_s). Therefore, the current-voltage (I-V) characteristic of the whole module can be derived by:

$$I = I_{ph} - I_o\left\{e^{\left[\frac{q}{n_s AKT}(V + n_s I R_s)\right]} - 1\right\} - \frac{V + n_s I_c R_s}{n_s R_{sh}} \qquad (4)$$

Where I is the module current and V is the module voltage. Now, the module output power (P) can be determined simply from

$$P = V.I \qquad (5)$$

3. Problem Formulation

According to the PV model, the main target is to determine the optimum parameters of the equivalent circuit model which can satisfy both current-voltage and power-voltage curves using different evolutionary algorithms.

Therefore, the optimization problem is to estimate the best values for the four PV parameters (R_s , R_{sh}, A, and I_o) such that the fitness function $J = f(R_s, R_{sh}, A, I_o)$ is minimized where ' R_s ' varies from ' $R_{s\,\min}$ ' to ' $R_{s\,\max}$', ' R_{sh} ' varies from ' $R_{sh\,\min}$ ' to ' $R_{sh\,\max}$', ' A ' varies from ' $A_{\min}$' to ' $A_{\max}$', and ' I_o ' varies from ' $I_{o\,\min}$' to ' $I_{o\,\max}$'.

In all algorithms the unified fitness (objective) function of equation (6) is selected to test the estimated parameters. The fitness (objective) function is to minimize J such that:

$$J = Sum\ (abs\ ((I_{a\_actual} - I_{a\_estimated}) \qquad (6)$$

4. Evolutionary Algorithms

Several optimization techniques have been developed during the last two decades. GA is a commonly known evolutionary computational method that is inspired from the Darwinian theory of evolution in natural biology [10].

On the other hand, relying on the social behavior of swarm of bees, fish and other animals, the concept of the PSO has been developed [11].

Other modern algorithm, BFO is a heuristic search technique that was developed based on modeling of bacteria *E. coli* behavior present in human intestine and it has been proven that it is

efficient for various engineering optimization problems [12].

SAO is a different algorithm based on the analogy between the annealing process in metallurgy where heating and controlled cooling of materials is used to recrystallize metals by increasing the temperature to the maximum values until the solids almost melted then decreasing the temperature slowly until the particles are arranged and the system energy becomes minimal [13]. More details about the above mentioned optimization algorithms are presented in the following sections.

4.1 Genetic Algorithm (GA)

4.1.1 Overview of GA

GA is essentially a method to generate a new population or generation from a given population. In this process the selection, crossover, and mutation operators are being used. Each member of the population, called a chromosome, is a possible solution for the problem under consideration, and is represented as a binary chain. Members of each generation are ranked according to a specific criterion called fitness.

The choice operator gives those members a higher fitness ranking a better chance of being present in the next generation. The crossover and mutation operators are applied to each chromosome with a specific probability and cause new chromosomes to be present in the new generation.

To solve the estimation problem using GA, all possible solutions have to be coded in chromosomes; that is the four parameters of the PV module. Series binary coding is used in this paper. Next, to calculate the fitness of a chromosome, the optimization function has to be calculated using the information in the chromosome.

Therefore, the GA begins, like any other optimization algorithm, by defining the optimization variables, the fitness function, and the fitness. It ends like other optimization algorithms too, by testing for convergence.

4.2.2 Steps of GA Algorithm

The general steps implemented when using GA are:

1. Generate a random initial population.
2. Create the new population by applying the selection and reproduction operators to select pairs of strings. The number of pairs will be the population size divided by two, so the population size will remain constant between generations.
3. Apply the crossover operator to the pairs of the strings of the new population.
4. Apply the mutation operator to each string in the new population.
5. Replace the old population with the newly created population.
6. Copy the best-fitted individuals to the newly created population to warrantee evolution.
7. If the number of iterations is less than the maximum go to step two, else stop. Or, if the fitness of the best result does not get better over certain number of iterations, then stop.

4.2 Particle Swarm Optimization (PSO)

4.2.1 Overview of PSO

Particle swarm optimization (PSO) is an optimization technique capable of finding global optimal points by using the social interaction of unsophisticated agents. In PSO, a population of particles flies through a search space with velocity updated by movement inertia, self cognition, and social interaction. Initially, the particles are randomly placed in a search space of a certain problem or function. So, each particle in swarm represents a solution to the problem. An objective function is evaluated for each particle. For each particle, its velocity and position is updated at each iteration by the following equations:

$$v_i(k+1) = w\,v_i(k) + c_1 r_1 \left[L_i(k) - x_i(k)\right] + c_2 r_2 \left[G(k) - x_i(k)\right] \quad (7)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (8)$$

Where x_i is the position of the *i-th* particle in the search space, v_i is the velocity of the *i-th* particle, w is particle inertia, c_1 is cognitive acceleration constant, c_2 is social acceleration constant, L_i is particle's best known position, G is the best known position found by all particles, and r_1, r_2 are random number between 0 and 1.

According to equations (7) and (8), each particle is updated by two best values, *L* and *G*. *L* is the local best solution that the particle achieved so far. *G* is the global best solution obtained so far by any particle within the neighborhood. Large values of *w* favor higher ability for global search, while low values of *w* favor a higher ability for local search.

4.2.2 Steps of PSO Algorithm

The PSO algorithm is divided into the following steps:

1. Initialization: In this step, the PSO parameters are initialized randomly.
2. Evaluation of the initial position: where the cost for all particles in the initial population are evaluated according to the objective function.
3. Updating position and velocity: The velocity and position of the particles are updated according to equations (7) and (8).
4. Evaluation of the updated position: The updated positions of particles are evaluated and the local best *L* and global best *G* particles are updated.
5. Check if the terminal condition is satisfied: If the terminal condition has not been satisfied, the updating process will be repeated; otherwise, the optimization process ends. The terminal condition is a maximum number of iterations.
6. Output results: The best solution *G* is obtained during the optimization.

4.3 Bacterial Foraging Optimization (BFO)

4.3.1 Overview of BFO

BFO is an algorithmic approximation technique mimicking bacteria colony growth. BFO is a non gradient optimization problem which is inspired by the foraging strategy used by *E. coli* bacteria such that it maximizes their energy intake (*E*) per unit time (*T*) spent in foraging. The four principal mechanisms observed in bacteria are chemotaxis, swarming, reproduction, and elimination-dispersal [14-15].

Chemotaxis

The movement of *E. coli* bacteria in the human intestine in search of nutrient-rich location away from noxious environment is accomplished with the help of the locomotory organelles known as flagella by chemotactic movement in either of the ways, that is, swimming or tumbling.

Suppose $\theta^i(j, k, l)$ represents the i^{th} bacterium at j^{th} chemotactic, k^{th} reproductive, and l^{th} elimination-dispersal step.

Then chemotactic movement of the bacterium may be mathematically represented by equation (9). In which, $C(i)$ is the size of the unit step taken in the random direction, and $\Delta(i)$ indicates a vector in the arbitrary direction whose elements lie in the range $[-1,1]$ as follows:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\times\Delta(i)}} \quad (9)$$

Swarming

This group behavior is seen in several motile species of bacteria, where the cells, when stimulated by a high level of *succinate*, release an attractant *aspertate*. This helps them propagate collectively as concentric patterns of swarms with high bacterial density while moving up in the nutrient gradient. The cell-to-cell signaling in bacterial swarm via attractant and repellant may be modeled as per (10), where $J_{cc}(\theta(i, j, k, l))$ specifies the objective function value to be added to the actual objective function that needs to be optimized, to present a time varying objective function, *S* indicates number of bacteria in the population, *p* is the number of variables to be optimized, and $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^T$ is a point in the *p*-dimensional search domain. The coefficients $d_{\text{attractant}}$, $w_{\text{attractant}}$, $h_{\text{repellant}}$ and $w_{\text{repellant}}$ are the measure of quantity and diffusion rate of the attractant signal and the repellant effect magnitude, respectively,

$$J_{cc}\left(\theta(i,j,k,l)\right) = \sum_{i=1}^{S} J_{cc}\left(\theta, \theta^i(i,j,k,l)\right) = \sum_{i=0}^{S}(-d_{attractant}\ exp(-w_{attractant}\ \sum_{m=1}^{P}(\theta_m - \theta_m^i)^2\ )) + \sum_{i=1}^{S}(h_{repellant}\ exp(-w_{repellant}\ \sum_{m=1}^{P}(\theta_m - \theta_m^i)^2)) \quad (10)$$

Reproduction and Elimination-Dispersal

The fitness value for i^{th} bacterium after travelling N_c chemotactic steps can be evaluated by the following equation:

$$J^i_{healt\ h} = \sum_{j=1}^{N_c+1} J^i(j,k,l) \quad (11)$$

Here $J^i_{healt\ h}$ represents the health of i^{th} bacterium. The least healthy bacteria constituting half of the bacterial population are eventually eliminated while each of the healthier bacteria asexually split into two, which are then placed in the same location. Hence, ultimately the population remains constant.

4.3.2 Steps of BFO Algorithm

The BFO algorithm is divided into the following steps:

1. Initialize the parameters
   a. Initialize the counters that is, chemotactic loop counter (*j*), reproduction loop counter (*k*), elimination dispersal loop counter (*l*), swim

counter (*m*) to zero and set bacterium index *i* = 0

2. Elimination–dispersal loop *l=l+1*
3. Reproduction loop *k=k+1*
4. Chemotaxis loop *j=j+1*
   a. For i=1,2,3,…,S take a chemotactic step for bacterium i as follows
   b. Compute value of objective function J_{cc} .
   c. Tumble**:** Generate a random vector with each element a random number in the range [−1, 1] as per (9),
5. Calculate the health of each bacterium.
   a. Discard *Sr* number of bacteria with least health
6. Compute and update J_{cc} as per (10).
7. Calculate the new objective function $J(i, j + 1, k, l)$
8. Swim
   a. Let m=0 (counter for swim length)
   b. While m < Ns (if have not climbed down too long)
   c. Let m = m+1
   d. If $J(i, j + 1, k, l) < J_{\text{last}}$ (if doing better), set $J_{\text{last}} = J(i, j + 1, k, l)$
   e. Else m < Ns Go to next bacterium (i + 1).
9. If $J < N_C$, go to step 4. In this case, continue chemotaxis since the life of the bacteria is not over.
10. Reproduction
    For the given k and l, and for each, let i=1, 2, 3,….S update equation (3).
11. If we have not reached the number of specified reproduction steps, so start the next generation of the chemotactic loop and perform Elimination-Dispersal For with probability, eliminates and disperses each bacterium (this keeps the number of bacteria in the population constant) To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain or search space.
    If $l < N_d$, then go to step 1; otherwise end.

4.4 Simulated Annealing Optimization (SAO)

4.4.1 Overview of SAO

Annealing is the process of heating the solid body to a high temperature and allowed it to cool slowly causing the particles of the solid material to reach the minimum energy state. The mathematical equivalence of the thermodynamic annealing as described above is called simulated annealing.

The energy of the particle in thermodynamic annealing process is corresponding to the cost function to be minimized in optimization problems. Initially the values assigned to the variables are randomly selected from a wide range of values. The cost function corresponding to the selected values are treated as the energy of the current state.

Searching the values from the wide range of the values can be compared with the particles flowing in the solid body when it is kept in high temperature.

The next energy state of the particles is obtained when the solid body is slowly cooled. This is equivalent to randomly selecting next set of the values. When the solid body is slowly cooled, the particles of the body try to reach the lower energy state. However, as the temperature is high, random flow of the particles still continues and hence there may be chance for the particles to reach higher energy state during this transition.

Probability of reaching the higher energy state is inversely proportional to the temperature of the solid body at that instant. The values are randomly selected so that cost function of the currently selected random values is minimum compared with the previous cost function value. At the same time, the values corresponding to the higher cost function compared with the previous cost function are also selected with some probability.

The probability depends upon the current simulated temperature ‘T’. If the temperature is large, probability of selecting the values corresponding to higher energy levels are more. This process of selecting the values randomly is repeated for a finite number of iterations. The values obtained after the finite number of iterations can be assumed.

4.4.2 Steps of SAO Algorithm

The SAO can be summarized as follow:

1. Initialize the value of the temperature ‘T’.
2. Randomly select the current values for the variables R_s , R_{sh}, A, and I_o from their allowable ranges. Let them be R_{sc} , R_{shc}, A_c, and I_{oc} respectively.
3. Compute the corresponding cost function value $f($ R_{sc} , R_{shc}, A_c, I_{oc}) .
4. Randomly select the next set of values for the

variables R_s, R_{sh}, A, and I_o from their allowable ranges. Let them be R_{sn}, R_{shn}, A_n, and I_{on} respectively.

5. Compute the corresponding cost function value $f(R_{sn}, R_{shn}, A_n, I_{on})$.
6. If $f(R_{sn}, R_{shn}, A_n, I_{on}) <= f(R_{sc}, R_{shc}, A_c, I_{oc})$, then the current values for the random variables $R_{sc} = R_{sn}$, $R_{shc} = R_{shn}$, $A_c = A_n$ and $I_{oc} = I_{on}$
7. If $f(R_{sn}, R_{shn}, A_n, I_{on}) > f(R_{sc}, R_{shc}, A_c, I_{oc})$, then the current values for the random variables $R_{sc} = R_{sn}$, $R_{shc} = R_{shn}$, $A_c = A_n$ and $I_{oc} = I_{on}$ are assigned when
   $exp\ [(f(R_{sc}, R_{shc}, A_c, I_{oc}) - f(R_{sn}, R_{shn}, A_n, I_{on}))\ / T] > rand$
   Note that when the temperature 'T' is less, the probability of selecting the new values as the current values is less.
8. Reduce the temperature T = r X T, where r is a scaling factor varying from 0 to 1.
9. Repeat steps 3:8 for n times until 'T' reduces to the particular value of 'T'.

5. Simulation Results

In this section, simulation results of PV module parameters estimation using the selected evolutionary algorithms are presented. In Table 1, exact and estimated PV parameters are presented. According to the results, PSO offers best estimated values. Then, BF offers also good estimated values. However, GA and SA estimated the parameters with lower accuracy.

Table 1 Estimated PV parameters with different techniques

M / P	Exact	PSO	BF	SA	GA
R_S	0.41138	0.4104	0.411	0.412	0.425
R_P	150	149.976	149.95	149.367	149.85
I_o	2.35e-8	2.35e-8	2.38e-8	2.18e-8	2.96e-8
A	1.21	1.21	1.211	1.205	1.226
Obj_ Fun.	0	4.371e-11	0.0000348	0.13342	1.0297

In Table 2, the computed PV parameters based on the estimated values of Table 1 are depicted. The computed relative errors are also presented in the same table. The results proved the capability of the evolutionary methods to estimate accurately the PV parameters with different tolerances. The best method was the PSO, while the relatively worst method was the GA.

Table 2 Computed PV parameters with different techniques

Method	P_{MAX} (watt)	I_{MAX}	V_{MAX}	I_{SC}	V_{OC}
EXACT	72.3296	4.3572	16.6	4.7869	21.389
PSO % error	72.3296 0 %	4.3572 0 %	16.6 0 %	4.7869 0 %	21.389 0 %
BF % error	72.3295 0%	4.3572 0 %	16.6 0 %	4.7869 0%	21.3 0.4161%
SA % error	72.6612 0.45 %	4.3772 0.45 %	16.6 0 %	4.7868 0.002 %	21.3 0.4161%
GA % error	71.9862 0.4748 %	4.3365 0.4751 %	16.6 0%	4.7864 0.01%	21.4 0.0514%

The estimated parameters with each method are utilized to plot both I-V and P-V curves as shown in Figures 2, 3, 4 and 5.The I-V and P-V curves with PSO are plotted in Fig.2.a and Fig.2.b, respectively.

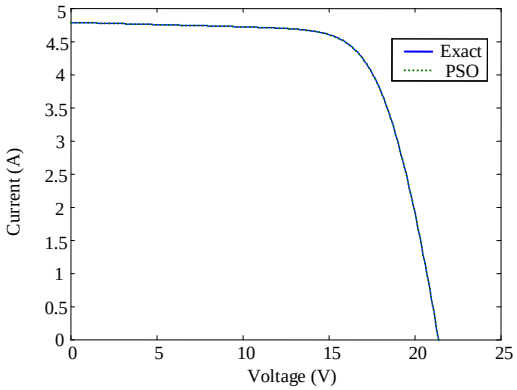

Fig.2.a, I-V curves with PSO

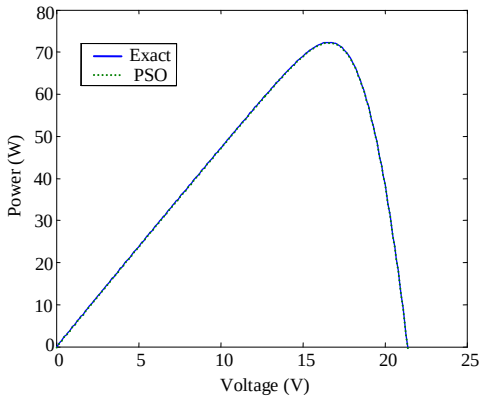

Fig.2.b, P-V curves with PSO

The I-V and P-V curves with BF are plotted in Fig.3.a and Fig.3.b, respectively.

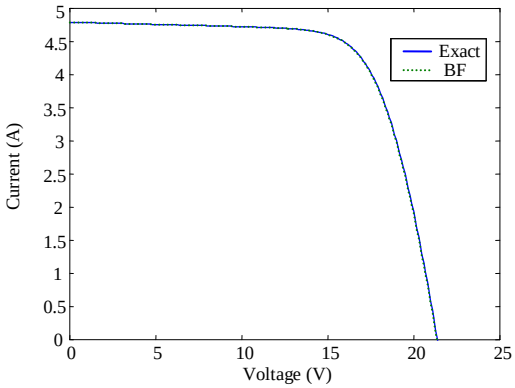

Fig.3.a, I-V curves with BF

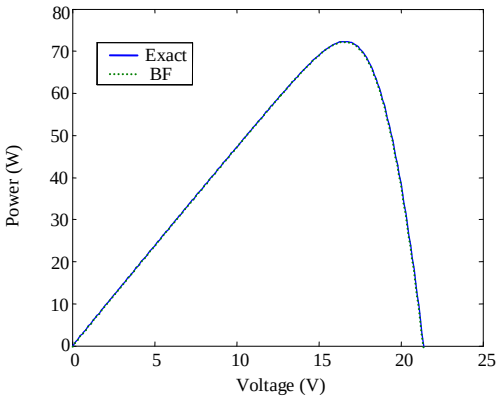

Fig.3.a, P-V curves with BF

The I-V and P-V curves with SA are plotted in Fig.4.a and Fig.4.b respectively.

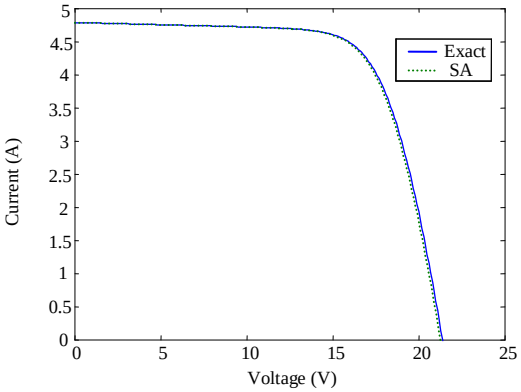

Fig.4.a, I-V curves with SA

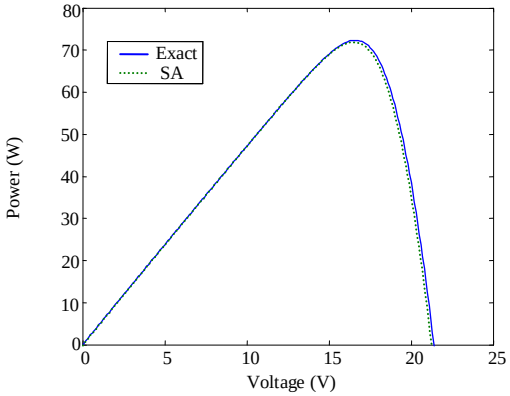

Fig.4.b, P-V curves with SA

The I-V and P-V curves with GA are plotted in Fig.5.a and Fig.5.b respectively.

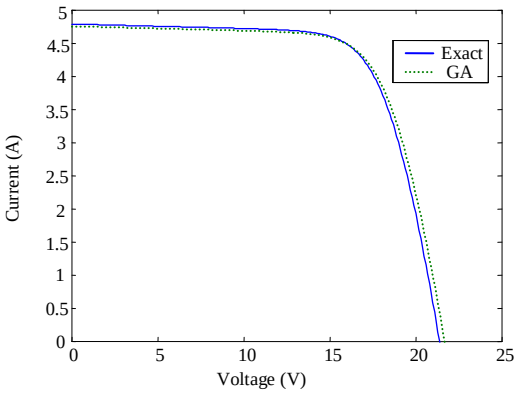

Fig.5.a, I-V curves with GA

The differences between actual values of PV current and estimated values are plotted in Fig.6.b for all algorithms. Moreover, the corresponding difference in PV power between actual values and estimated values are also plotted in Fig.6.b.

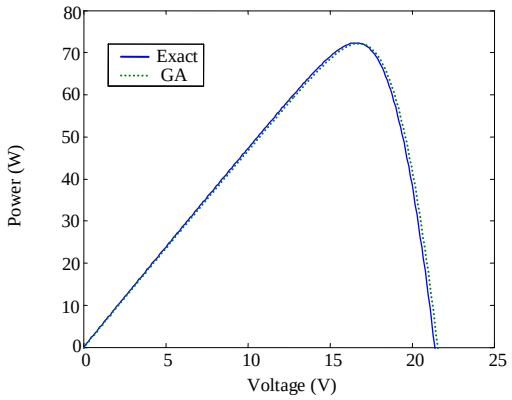

Fig.5.b, P-V curves with GA

According to the obtained curves, PSO gives the best performance, then BF, then SA and the worst performance was of GA.

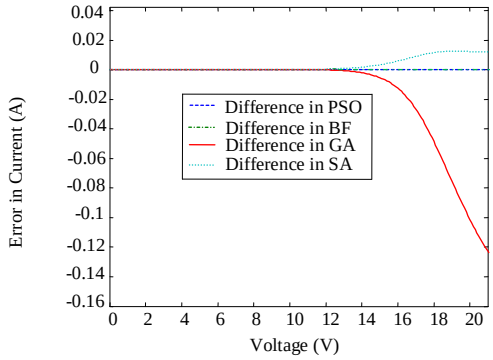

Fig.6.a, Difference between actual and estimated PV current

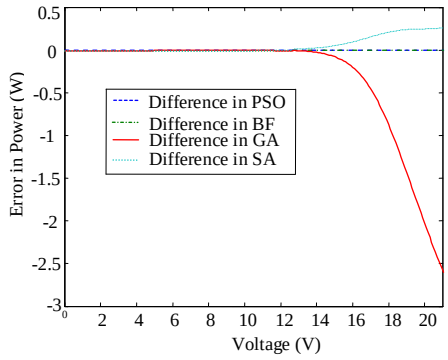

Fig.6.b, Difference between actual and estimated PV power

The following graphs illustrate the evolution of objective function with each algorithm. According to the results, PSO reached to the minimum value (4.371e-11) after 380 iterations approximately as shown in Fig.7.a. BF reaches a minimum value of (0.0000348) after 200 iterations as indicated in Fig.7.b. Moreover, the SA reached to the minimum value (0.13342) after 230 iterations approximately as shown in Fig.7.c. Finally, GA reached to the minimum value (1.0297) after 500 iterations approximately as shown in Fig.7.d.

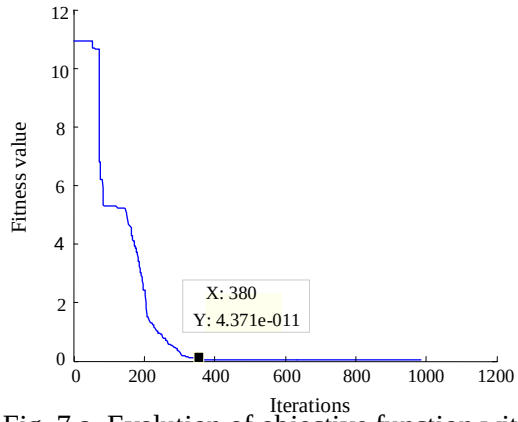

Fig. 7.a, Evolution of objective function with PSO

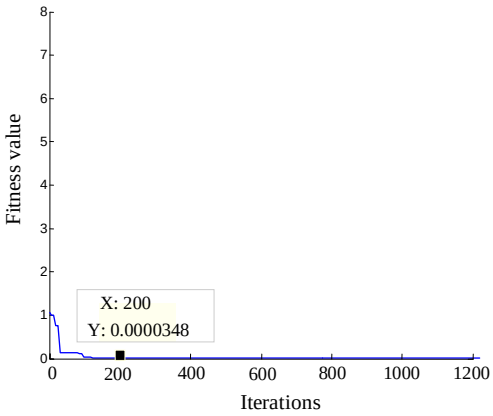

Fig. 7.b, Evolution of objective function with BF

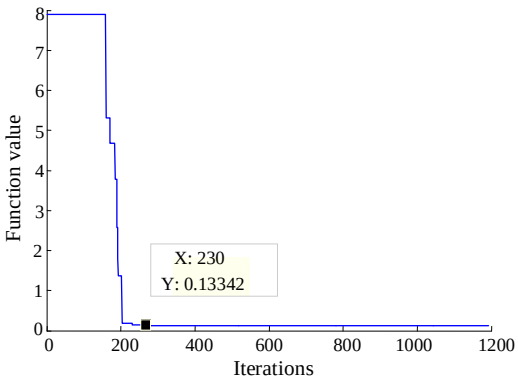

Fig. 7.c, Evolution of objective function with SA

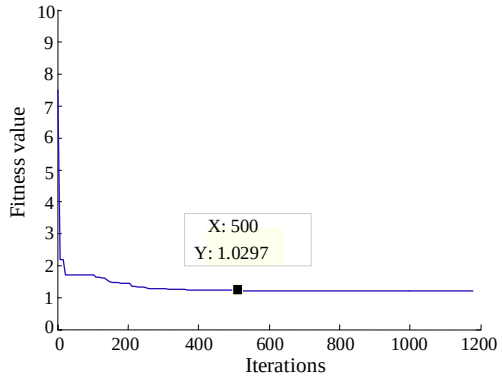

Fig. 7.d, Evolution of objective function with GA

6. Conclusion

In this paper several evolutionary algorithms (Particle Swarm, Bacterial Foraging, Simulated Annealing and Genetic Algorithms) have been applied to estimate the parameters of a real PV module. The comparison between these algorithms shows that PSO is the most efficient method to solve the optimization problem in terms of accuracy and convergence steps. Among these methods, SA exhibit relatively lower accuracy and slower convergence speed, while BF has better accuracy. Compared to other algorithms, GA has the worst accuracy and slowest convergence speed which makes GA is not adequate algorithm to solve such optimization problems. In general, evolutionary computational methods are able to estimate the PV parameters with acceptable precision.

References

1. Abir, C., Ali, K, Dhruv, K.: *Identification of Photovoltaic Source Models*. In: IEEE Transactions on Energy Conversion, Vol. 26 (2011), No.3, September 2011, p. 883 – 889.
2. Mohiuddin A., AI-Ahsan T., Mahmuda A. T.: *Estimation of Important Parameters of Photovoltaic Modules from Manufacturer's Datasheet.* In: IEEE/OSA/IAPR International Conference on Informatics, Electronics & Vision‘12 , 2012, p. 571-576.
3. Filippo, A., Attilio, D., Mario, S., Maurizio, S.*: Uncertainty Analysis in Photovoltaic Cell Parameter Estimation.* In: IEEE Transactions on Instrumentation and Measurement, Vol. 61, (2012), No. 5, May 2012, p. 1334-1342.
4. Loredana C., Marco F., Marco R., Sergio T.: *A Simplified Model of Photovoltaic Panel*. In: Instrumentation and Measurement Technology Conference (I2MTC) ‘12, 2012, p. 431-436.
5. Xue L., Sun L., Huang W., Jiang C.: *Solar Cells Parameter Extraction Using a Hybrid Genetic Algorithm. In:* Third International Conference on Measuring Technology and Mechatronics Automation ‘11, 2011, p. 306-309.
6. Kashif I., Zainal S., Hamed T., Amir S.: *Parameter Extraction of Photovoltaic Cell Using Differential Evolution Method*. In: IEEE Applied Power Electronics Colloquium (IAPEC) ‘11, 2011, p. 10-15.
7. Hengsi Q. , Jonathan W.: *Parameter Determination of Photovoltaic Cells from Field Testing Data using Particle Swarm Optimization*. In: Power and Energy Conference at Illinois (PECI) ‘11, 2011, p. 1-4.
8. Jiang C., Xue L., Song D., and Wang J.: *Solar Cells Performance Testing and Modeling Based on Particle Swarm Algorithm. In:* International Conference on Computer Science and Information Processing (CSIP) ‘12, 2012.
9. Eftichios, K., Kostas, K. , Nicholas, C.: *Development of a Microcontroller-based, Photovoltaic Maximum Power Point Tracking Control System*. In: IEEE Transactions on Power Electronics, Vol. 16 (2001), No. 1, January 2001, p. 46-54.
10. Moldovan N., Picos R., Garcia-Moreno E.: *Parameter extraction of a solar cell compact model using genetic algorithms*. In: Proceedings of the 2009 Spanish Conference on Electron Devices‘09, 2009, p. 379-382.
11. Kennedy J., and Eberhart R.: *Particle Swarm Optimization*. In: Proceeding of the IEEE International Conf. on Neural Network ’95, 1995, p. 1942-1948.
12. Zhang Y. and Wu L.: *Bacterial Chemotaxis optimization for protein folding model*. In: 5th International Conference on Natural Computation, ICNC‘09, 2009, p. 159-162.
13. Gopi E. S.: Algorithm Collections for Digital Signal Processing Applications Using Matlab, P.O. Box 17, 3300 AA Dordrecht, Netherlands.
14. Biswas, A., Das, S., Abraham, A., Dasgupta, S.: *Analysis of the reproduction operator in an artificial bacterial foraging system* In: Applied Mathematics and Computation, vol. 215 (2010), , p. 3343-3355.
15. Hooshmand, R. A., Parastegari, M., Morshed, M. J.: *Emission, reserve and economic load dispatch problem with non-smooth and non-convex cost functions using the hybrid bacterial foraging-Nelder–Mead algorithm.* In: Applied Energy, Vol. 89 (2012), p. 443-453.