Solar Power Plant Optimization

Carlos Sanchez Reinoso
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina & National University of San Juan (UNSJ), Argentina

Román Buitrago
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina & UNL-CONICET, Argentina

Diego H. Milone
UNL-CONICET, Argentina

ABSTRACT

The objective of this chapter is to optimize the photovoltaic power plant considering the effects of variable shading on time and weather. For that purpose, an optimization scheme based on the simulator from Sanchez Reinoso, Milone, and Buitrago (2013) and on evolutionary computation techniques is proposed. Regarding the latter, the representation used and the proposed initialization mechanism are explained. Afterwards, the proposed algorithms that allow carrying out crossover and mutation operations for the problem are detailed. In addition, the designed fitness function is presented. Lastly, experiments are conducted with the proposed optimization methodology and the results obtained are discussed.

1. INTRODUCTION

In the photovoltaic conversion chain, modules are responsible for converting energy from the sun into electric power, and several researchers attempted to improve the performance of the modules and their cells. This is a possible approach to optimize the performance of the photovoltaic system. However, in a photovoltaic system there are other devices that make up the conversion chain. Interaction among these stages should be considered in the system optimization, which gains fundamental importance when increasing its size. A problem associated with photovoltaic power plants is shading. There are some studies that discuss its impact on photovoltaic systems (Martínez-Moreno, Muñoz, & Lorenzo, 2010) (Sullivan, Awerbuch, & Latham, 2011) (Ubisse & Sebitosi, 2009). In these systems, the maximum power point tracking stage is very sensitive to shading, thus affecting the magnitude and morphology of the output curve (Petrone, Spanuolo, & Vitelli, 2007). To a large extent, this is due to the fact that Maximum

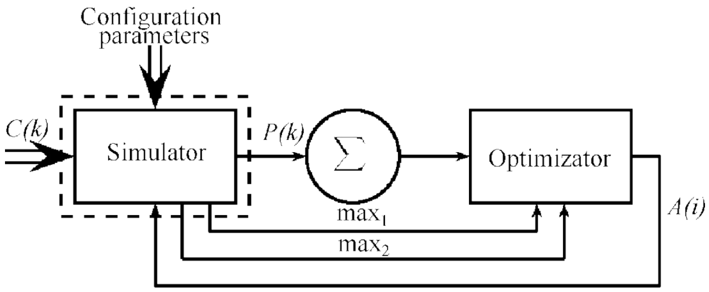
DOI: 10.4018/978-1-4666-6631-3.ch011

Power Point Tracking (MPPT) algorithms are usually based on the assumption that the generated power curve has only one peak (Houssamo, Locment, & Sechilariu, 2010) (Moradi & Reisi, 2011) (Enrique, Durán, Sidrach de Cardona, & Andújar, 2010) (Esram & Chapman, 2007). Sanchez Reinoso, Milone, and Buitrago (2013) introduces the different aspects mentioned previously within the simulation, and the heterogeneity of static and variable cloud cover on the performance of the system is taken into account. The said paper does an exploratory analysis based on the simulation of the effects of shading, but it does not propose any methodology to optimize the system. The way of designing a scheme that makes the most of the energy received, in the case of highly-powerful plants under shading, is scarcely studied. On the other hand, it is known that there exist optimization problems that demand to explore a wide range of solutions, making the classic methods almost inapplicable.

In recent years, different approaches to deal with these problems have been proposed, for example, evolutionary algorithms (EA). Some interesting features from this technique are the simplicity of the operators used, the possibility of using fitness functions with very few formal requirements, and the ability to explore multiple points of the search space in every iteration (Sivanandam & Depa, 2008). In the photovoltaic area, different computational intelligence techniques were successfully applied on the detection of operation in island mode (Chao, Chiu, Li, & Chang, 2011), on radiation modeling (Koca, Oztop, Varol, & Koca, 2011) (Ozgoren, Bilgili, & Sahin, 2012) (Mellit, Mekki, Messai, & Kalogirou, 2011), and on simulators of autonomous systems (Mellit, Mekki, Messai, & Salhi, 2010b) (Gómez-Lorente, Triguero, Gil, & Espín Estrella, 2012). In Larbes, Ait Cheikh, Obeidi, and Zerguerras (2009) and Messai, Guessoum, and Kalogirou (2011), evolutionary algorithms are employed to optimize the maximum power point tracker controller, whereas in Mellit, Kalogirou, and Drif (2010a) the dimension- ing of little isolated systems including batteries is optimized. In the case of grid-connected systems, the system optimization was also approached from the dimensioning point of view but using particle swarm optimization (Kornelakis & Marinakis, 2010). Other study that optimizes through evolutionary algorithms (Díaz-Dorado, Suárez-García, Carrillo, & Cidrás, 2011) focuses on the projection of shades among the installation components when using tracking mechanisms. It is noted in literature that although the effect shading has on photovoltaic systems has been proved, most optimization works focus on dimensioning (mainly, the array-inverter relationship), on static shades and on improving the maximum power point tracking algorithms. However, if the first stage is not optimized, the following stages, at best, would only be able to mitigate greater losses.

This chapter proposes a methodology that makes possible to consider variable shading and optimize photovoltaic power plants that can be composed of a great number of modules, considering the simulation of all phases in the conversion chain and basing the global optimization of the system on the array configuration. In particular, will be used genetic programming techniques and will be proposed different fitness functions and methods of initialization, crossover and mutation. The organization of the chapter is the following: next sections present the approach used in the optimization, the simulation, and there follows a description of the characteristics of the algorithm developed in the proposed optimization methodology. Results are presented and discussed in Section 3. Finally, conclusions are summarized and future works are proposed.

Figure 1. Block diagram of the optimization methodology



2. PROPOSED MODEL AND ALGORITHM

Figure 1 shows a general scheme of the methodology used in the problem optimization. The sequence of climatic data *C(k)* enters the deterministic simulator, which is parameterized through *m* parameters: number of modules, power of modules, power of the inverter, maximum voltage of the inverter, maximum current of the inverter, etc. Although the simulator allows obtaining several outputs, the one used in this optimization approach is power *P(k)*. This output is added for the whole sequence of length *l* and the result is one of the optimization block inputs. Other inputs from this optimizing block are internal parameters of the simulator such as maximum allowable voltages and currents. The optimizer is based on an evolutionary algorithm and those inputs are functionally related to make a cost function. In the iteration *i* of the global loop, the optimizer gives as an output a population of solutions A_i based on the cost function evaluation. Simulations are essentially based on the approach described in Sanchez Reinoso, Milone and Buitrago (2013).

The aim is to solve an optimization problem regarding the type and distribution of connections that are made on a set of solar panels, in order to obtain the best performance possible in heterogeneous situations. Two objective functions are proposed, which consider maximizing the generated accumulated power and minimizing the number of output curve peaks. The objective is to find the optimum configuration of the system *w**. This configuration is composed of a subset of connections and a subset of modules. The constraint established was that the cardinality of the configurations' connection set must be constant. That is to say that the initial number of photovoltaic modules must remain invariable, which not only implies avoiding the elimination and creation of new modules, but also not repeating them. In addition, voltages and currents entering the inverter must be confined to their respective allowable ranges.

Below, the representation of circuits will be covered. Next, the proposed mechanism for the initializa- tion of populations will be discussed. From Section 2.3 onwards, the proposed mechanisms for selec- tion, crossover and mutation will be covered in detail. Finally, the fitness function will be developed in Section 2.5.

2.1. Representation

A widely used representation in evolutionary computation is that of trees. This chapter attempts to solve a problem that leads us, almost naturally, to a tree representation. An electric connection system can be exactly represented through trees, using the roots as *type* of connection (series/parallel) and hosting the *elements* of the circuit in the leaves (photovoltaic solar modules, in this case). Figure 2 shows three cases of traditional representation of electric circuits and their respective tree representation. The first and second cases correspond to parallel module connections in one and two levels, respectively. The third case shows a series-parallel hybrid connection, where it is possible to observe how suitable the tree representation is since it makes possible to establish the appropriate precedence order. In addition, the said structure may be useful in subsequent studies, for instance, for analyzing solutions through metrics usually used in trees.

Figure 2. Traditional and tree representation for electric circuits

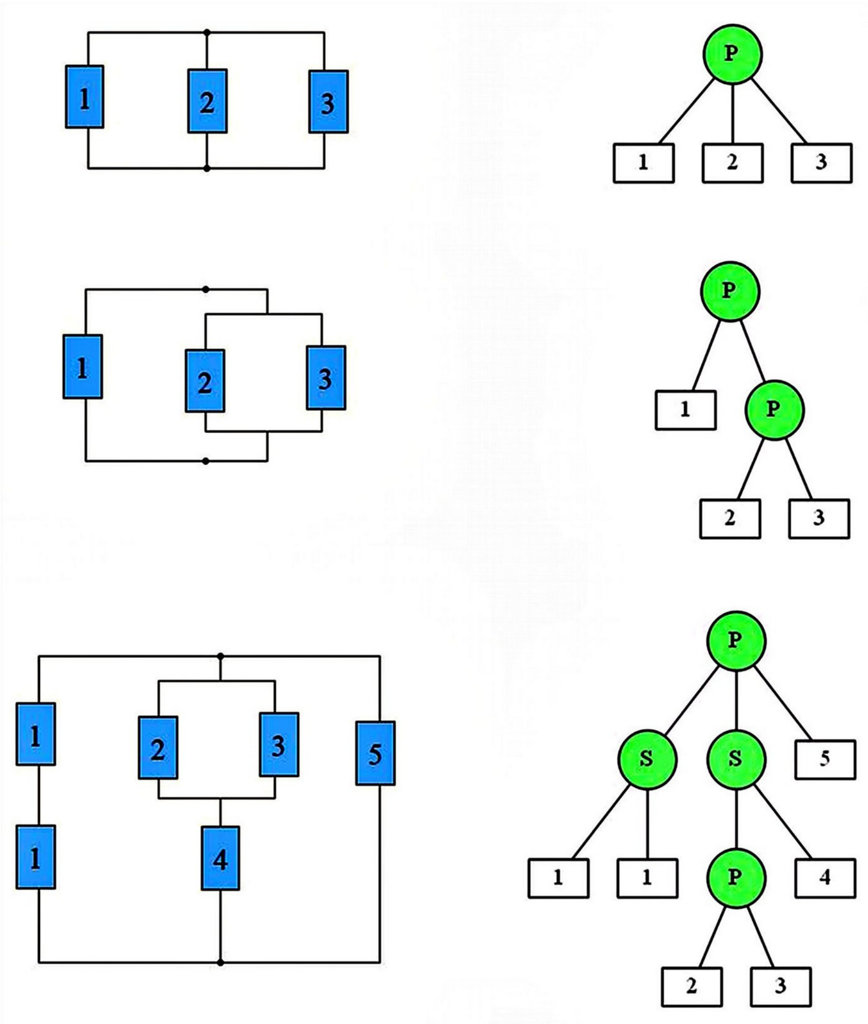
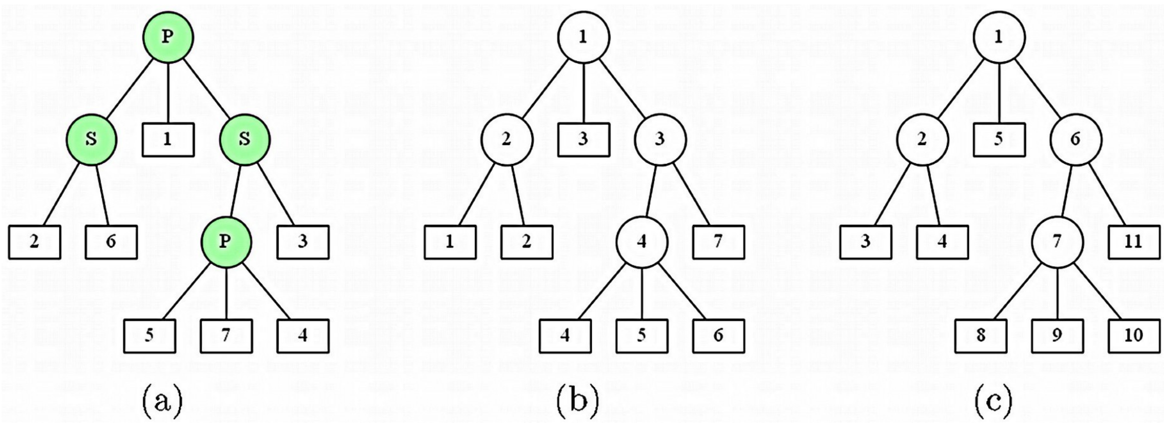


Figure 3. Ways of indexing nodes. (a) Graphic representation of a tree with 7 modules, (b) Relative indexing of (a), (c) Absolute indexing of (a)



In order to operate on these trees, two different ways of referencing (indexing) the elements of the structure were used: the relative and the absolute one. In the first one, roots and leaves have an independent index, whereas in the second, all nodes are listed without distinction of type. Both are based on going through the tree in pre-order (Root, Left child, Right child). Figure 3 shows an example of the indexes that correspond to each node of the tree for both indexing systems.

2.2. Initialization

Three types of initialization are proposed: deterministic, random and hybrid. The initial population is generated through algorithms that design connection trees, the constraint being that all of them must have K modules (the number of modules defined in the design is maintained) distributed in a rectangular array

Figure 4. Distribution of modules and connection trees for all modules connected in parallel and in series

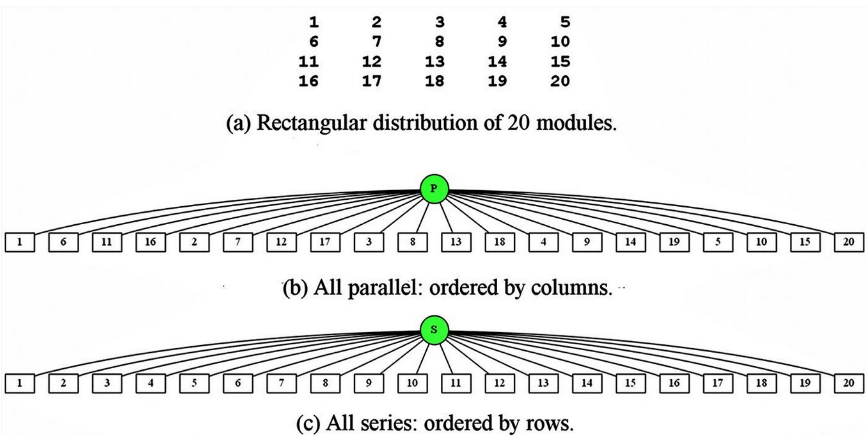
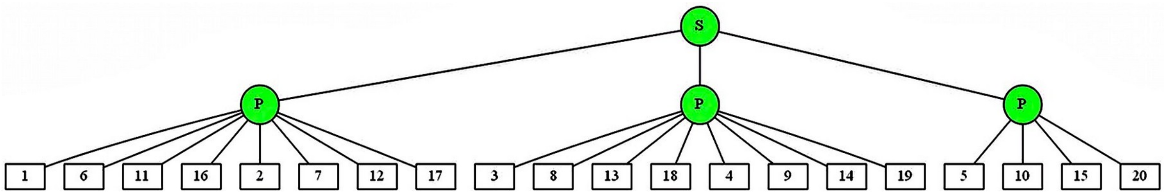


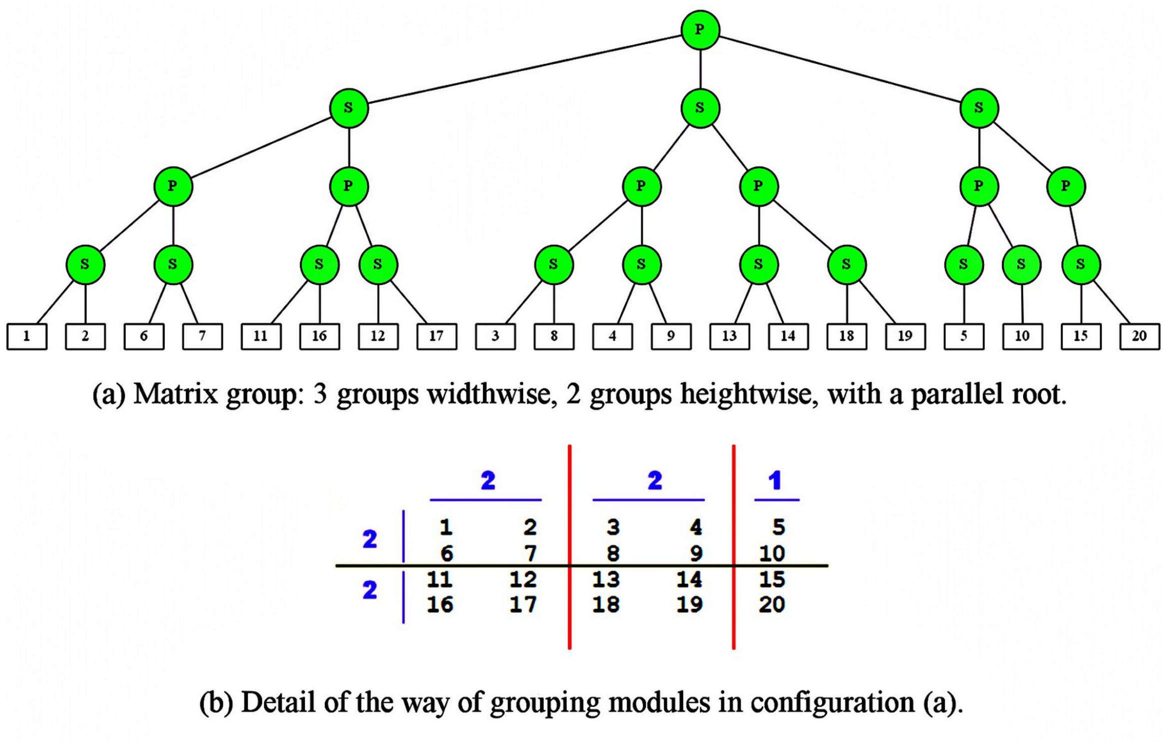
Figure 5. Simple group: 3 groups widthwise with a series root



(Figure 4(a)). In the deterministic initialization, series-parallel connections are made with subgroups according to known rules, which may be summarized as follows:

- All *P* or *S*: a single root connecting all modules in parallel or in series (Figure 4(b) and 4(c)).
- Simple groups: modules are grouped by one of the dimensions (width or height). The general configuration is inverse to the configuration of the groups. For example, series groups joined in parallel, and vice versa (Figure 5).
- Matrix groups: similar to the previous one, modules are grouped in both dimensions through a previous partitioning of the distribution matrix (Figure 6(b)). The general configuration is inverse to the configuration of the groups (Figure 6(a)).

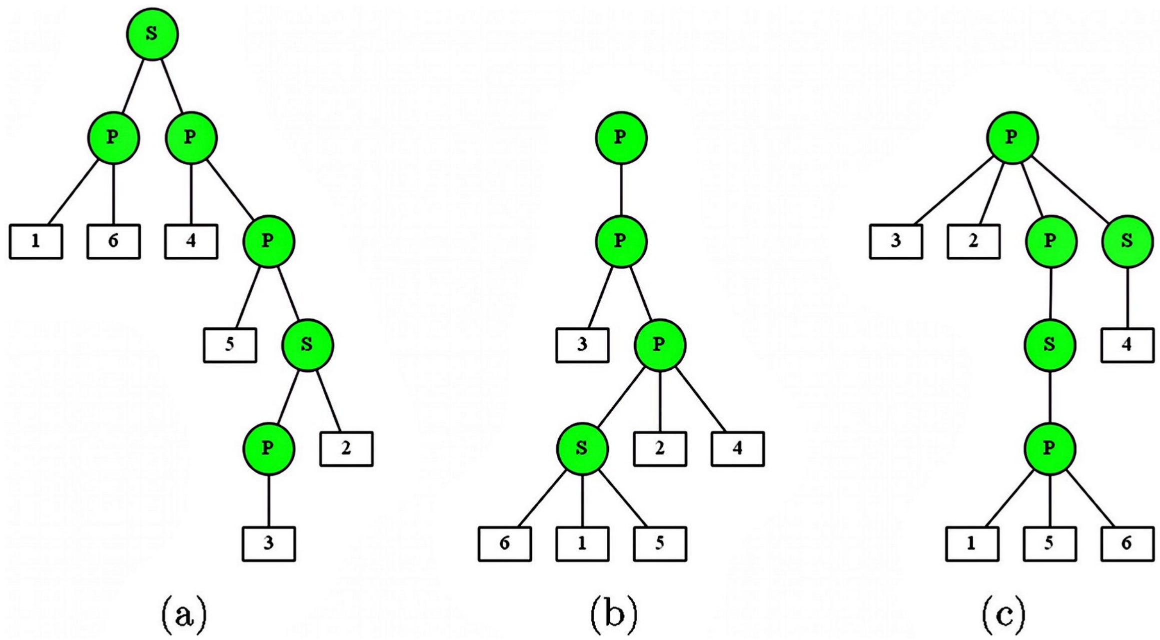
Figure 6. Matrix group type configurations and detail of the groupings



(a) Matrix group: 3 groups widthwise, 2 groups heightwise, with a parallel root.

(b) Detail of the way of grouping modules in configuration (a).

Figure 7. Three random configurations with 6 modules are shown in (a), (b) and (c)



On the other hand, random initialization is obtained through 30 random mutations (10 for each type of mutation described in 2.4.) of the elements of a deterministic tree of the matrix group type, whose parameters were set at random (Figure 7(a), 7(b), 7(c)). Whereas hybrid initialization combines a deterministic initialization with the random one (50% for each).

2.3. Crossover

Crossover of two connection trees has a significant constraint compared to the crossover methods usually used in genetic programming: the crossover must not modify the number of modules. For this purpose, a new crossover operator was designed, which is detailed in Algorithm 1 and represented graphically in Figure 8. First, all subtrees with an equal number of descendant modules in both parents are identified (Figure 8(a)). Then, a subtree of each parent is chosen at random (both with the same number of descen-
dants), which we will call $subtree_1$ and $subtree_2$ (Figure 8(b,c)). Subsequently, the list of descendants from $subtree_1$ is obtained and it is used to replace the descendants in $subtree_2$ (Figure 8(d)). At this point, $subtree_2$ keeps its structure and only the way the modules are connected varies. Finally, in $tree_1$ the $subtree_1$ is replaced by $subtree_2$ (Figure 8(e)). Consequently, $tree_1$ does not modify the number of modules and the set of module indexes is not altered (there are not repeated values and others are not eliminated).

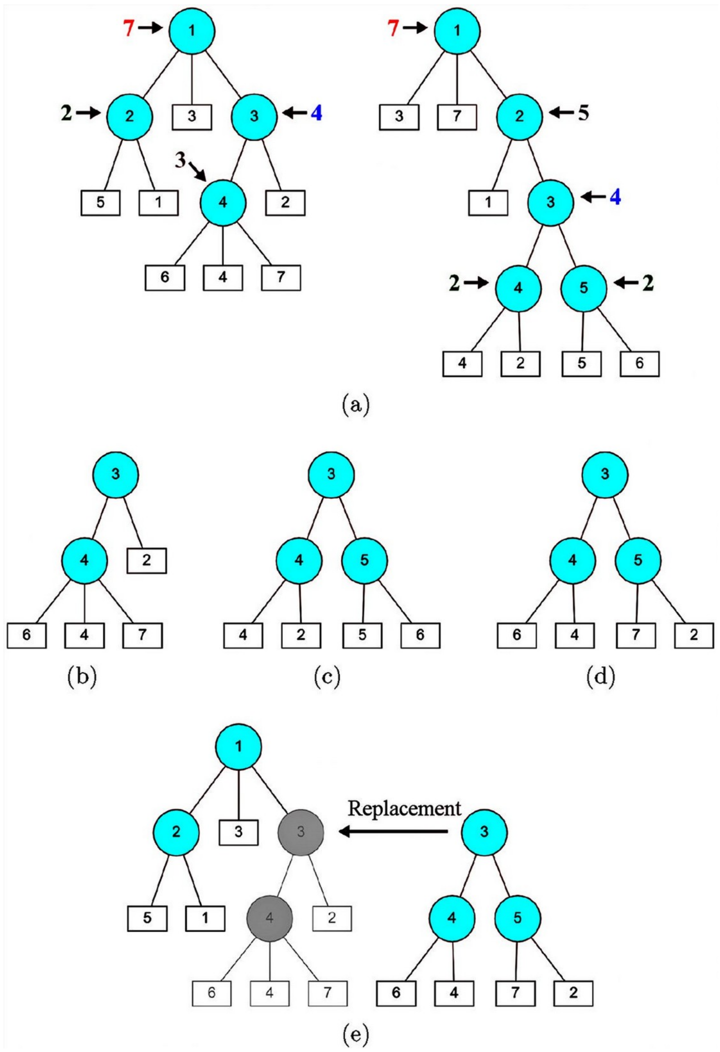
2.4. Mutation

Just as crossover, mutation must also be carried out in a way that the number of modules remains constant. Mutation operators capable of generating valid individuals were designed: node exchange operator, inversion operator for the type of connection and ungroup operator.

Algorithm 1. Crossover with constant modules

Input: configurations of $tree_1$ and $tree_2$.
Output: cross trees
1 For each root of $tree_1$ and $tree_2$ obtain the number of descendant modules, including all sublevels;
2 Determine compatible subtrees: roots of $tree_1$ that have a number of descendants equal to some root of $tree_2$;
3 List as possible crossover points roots with an equal number of descendants;
4 Choose a crossover point at random from the list of feasible points;
5 Obtain the list of descendants of $subtree_1$, with root in the crossover point of $tree_1$;
6 Take $subtree_2$ (root in the crossover point of $tree_2$) and replace the descendants with the original descendants of $subtree_1$;
7 Replace the crossover point of $tree_1$ with the updated $subtree_2$;

Figure 8. Crossover of configuration trees. (a) Number of descendants for each root, (b) $subtree_1$ (root 3), (c) $subtree_2$ (root 3), (d) $subtree_2$ with indexes from updated modules according to the order in $subtree_1$. (e) Replacement of $subtree_1$ through $subtree_2$



Exchange Operator

This operator modifies the structure of a tree exchanging two randomly selected nodes. Any element of the tree is considered as a node; it may be a connection (where the type is specified) or a leaf (where reference is made to a photovoltaic module). Therefore, when an operation is made with a connection node, the operation will be with the whole structure that descends from it. This independent manipulation is carried out using the absolute indexing to indicate the pair of nodes in the exchange.

Exchange operations may be grouped in three cases: Module-Module, Module-Connection, and Connection-Connection. The exchange, described by Algorithm 2, is quite complex since it must follow a coherent structure. Rules governing the exchange are:

- Two nodes not having any descendant relationship between them are free to exchange. When a node is of the Connection type, the exchange is made together with all of its descendants, maintaining the structure of that connection subtree.
- It is not possible to exchange a Connection for a Module within its descendants.
- If a Connection is exchanged for a Connection in its descendants, the first one is added as a direct descendant of the second one and the latter replaces it in the original tree.

Figure 9 shows the results of applying the different types of mutation by means of node exchange. The original tree and its absolute indexing are represented in Figure 9(a) and Figure 9(b), respectively. The result of the exchange of nodes 2 and 7 is an exchange of the corresponding subtrees (Figure 9(c)). Other possibility permitted by the proposed operator is the exchange of leaves, i.e. modules, as shown in Figure 9(d). When attempting to make an exchange between node 7 and 6, the descendant relationship implies separating the subtree with root in 7 from the subtree with index 6, assigning the subtree 6 as the child of 7, and inserting the new subtree 7 into the original place of the node with absolute index (Figure 9(e)). The operator also considers the possibility of exchanging a Connection type node and its subtree for a module (Figure 9(f)). Figure 9(g) shows the case of an exchange operation that cannot be made because a module cannot become the parent of any other element.

Inversion Operator

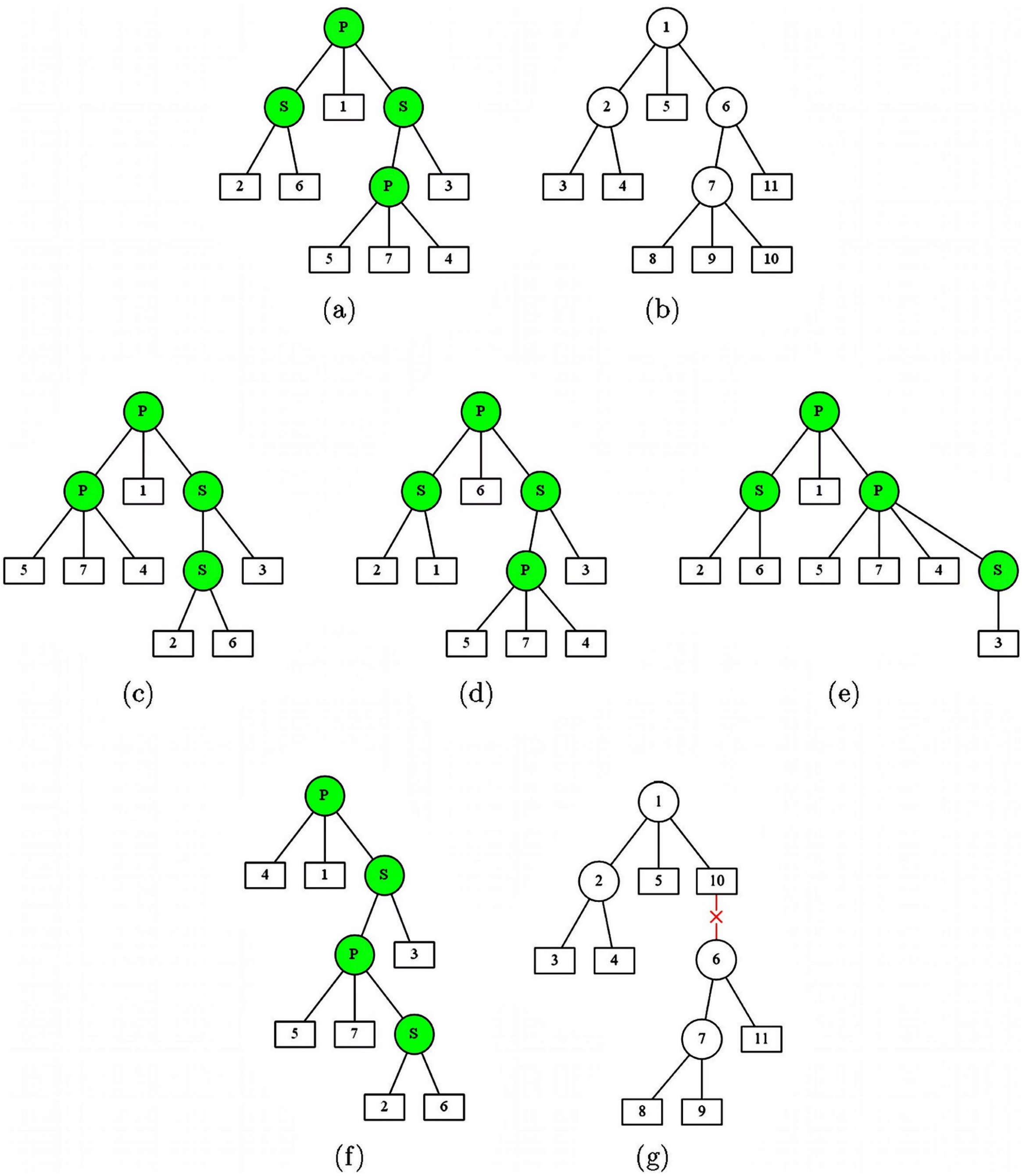
The inversion operator acts on the type of connection, without altering the connection direction of the components. Therefore, it is possible to mutate a root so that the type of connection changes from series to parallel, and vice versa. For that purpose, its value is located and updated using reference indexes relative to the type of connection.

Some examples of inversion of the tree from Figure 3(a) can be seen in Figure 10. In the case of Figure 10(a) and Figure 10(b), inversions of nodes 3 and 4 are made independently, whereas the remaining case (Figure 10(c)) corresponds to inverting node 3 and inverting node 4 to the result obtained.

Ungroup Operator

This mutation uses relative indexes and consists of ungrouping a single root from the original tree. The root is eliminated and the direct descendants are connected to the immediate higher root. The main root

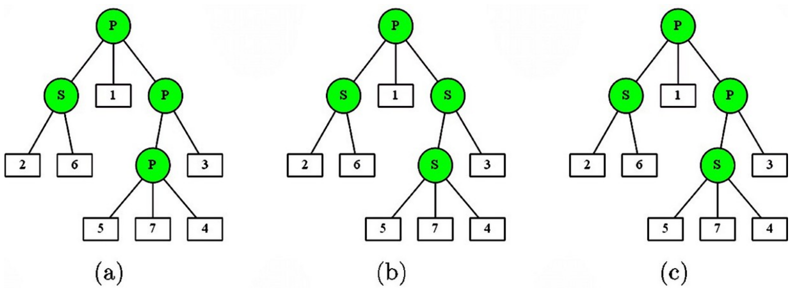
Figure 9. Examples of mutation by node exchange (absolute indexes are used). (a) Original configuration tree, (b) Absolute indexing, (c) Exchange 2-7, (d) Exchange 4-5, (e) Exchange 7-6, (f) Exchange 2-10,
(g) It is not possible to make the exchange 6-10 because a module cannot be a parent



Algorithm 2. Mutation by node exchange.

```
Input: configuration tree, absolute index_1 and index_2.
Output: configuration tree with exchanged nodes, or
    empty tree in case the exchange is not possible.
Exchange the indexes so that index_1 < index_2.
Obtain node_1 and node_2 as the elements of the original tree in index_1 and index_2, respectively.
if node_1 is a configuration then
  if node_2 is descendant of node_1 then
    if node_2 is a configuration then
      Eliminate node_2 from node_1
      Add node_1 as a child of node_2
      Replace tree in index_1 with node_2
    else
      tree = empty
    end if
  else
    Replace tree in index_2 with node_1
    Replace tree in index_1 with node_2
  end if
else
  Replace tree in index_2 with node_1
  Replace tree in index_1 with node_2
end if
return tree
```

Figure 10. Examples of mutation by inverting the type of connection (relative indexes are used, see Figure 3(b)). (a) Invert 3, (b) Invert 4, (c) Invert 3 and invert 4 to the result obtained



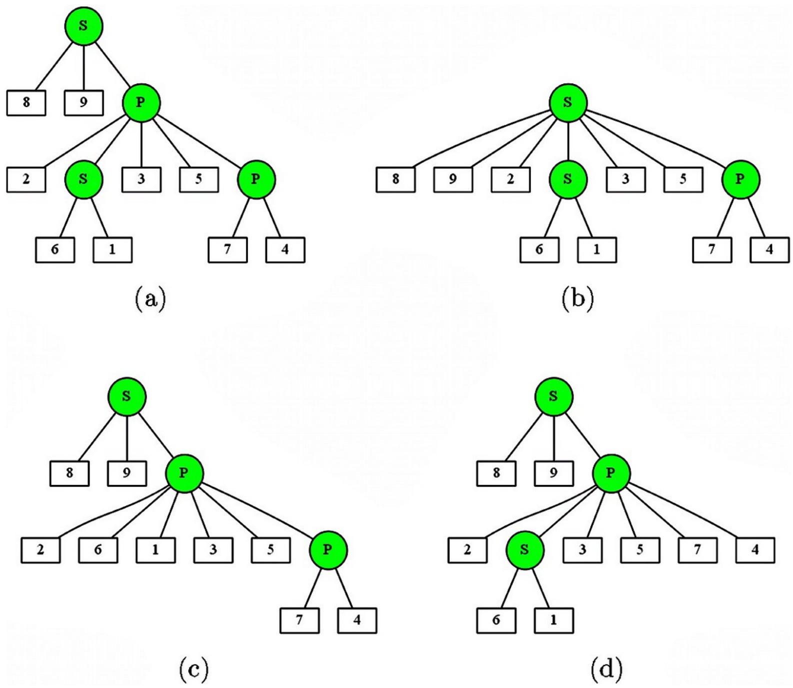
that originates the tree is the only one this technique cannot be applied to, since it would break the whole structure. Figure 11 presents examples of different ungrouping.

2.5. Fitness Function

The fitness function was designed to evaluate the quality of the individuals during the evolution and to conduct the search. This function shapes the characteristics the solution to the problem must have. The following aspects are considered:

- Relationship between the power given out by the array of modules and the nominal power of the array.
- Penalty for arrays that move away from allowable voltage and current limit values set by converters.

Figure 11. Examples of mutation by ungrouping (relative indexes are used). (a) Original configuration tree, (b) Ungroup root 2, (c) Ungroup root 3, (d) Ungroup root 4



The general evaluation equation has the following form:

$$f = \sum_{k=1}^{L} \frac{P(k)}{P_n}\Big(1 - g_1\big(I(k)\big) - g_2\big(V(k)\big) - g_3\big(P(k), V(k)\big)\Big) \tag{1}$$

where: L is the length of the sequence, P is the power, I is the current and V is the voltage. Constraints are represented through g_1, g_2 and g_3. The first two are functions that control the effect of their respective parameters, from which different methods are defined. On the other hand, function g_3 considers the multimodality of the output curve. P_n is the array nominal power, calculated as the number of modules by their nominal power.

This expression includes the optimality and feasibility of the problem. The power given out is related to the optimality of the problem, whereas the operating constraints that allow obtaining feasible solu- tions are the ones related to the allowable voltage and current ranges. Including the number of peaks is, in fact, an indirect way of adding a constraint in favor of the global optimality of the system.

Penalties

Different ways are proposed to penalize violations to the constraints. The common element among all of them is the use of variables that are calculated as the difference relative to the maximum allowable values

$$c_I = \left(I_{max} - I\right) / I_{max}$$

$$c_V = \left(V_{max} - V\right) / V_{max}$$

where it is observed that excesses over allowable limits cause c_I and c_V to result in negative values.

Fitness Functions with Penalties

Twelve different fitness functions that respond to the general form of (1) were implemented, using c_I and c_V:

Addition: the sum of coefficients causes excesses (negative values) to reduce fitness, whereas deficiencies are seen as fitness boosters:

$$g_1\left(I\right) = -c_I\left(I\right) \tag{2}$$

$$g_2\left(V\right) = -c_V\left(V\right) \tag{3}$$

$$g_3 = 0 \tag{4}$$

Absolute Addition: in this case, both defects and excesses are treated as penalties (they subtract) to the ideal fitness:

$$g_1\left(I\right) = \left|c_I\left(I\right)\right| \tag{5}$$

$$g_2\left(V\right) = \left|c_V\left(V\right)\right| \tag{6}$$

$$g_3 = 0 \tag{7}$$

Exponential: this function allows the fitness to penalize excesses and defects but with an exponential, non linear weight:

$$g_1\left(I\right) = \left|\left(e^{-c_I(I)} - 1\right)\right| \tag{8}$$

$$g_2(V) = \left|\left(e^{-c_V(V)} - 1\right)\right| \tag{9}$$

$$g_3 = 0 \tag{10}$$

Hyperbolic Tangent: in this case, a special function by parts is defined, which allows controlling different values for the excesses and defects. An important feature of this function is the possibility of controlling the degree and saturation of penalties. The hyperbolic tangent function can be defined as

$$tgH(x)\begin{cases} A_n(2(1/(1+e^{-m_n 2x}))) - A_n & si\ x < 0 \\ 0 & si\ x = 0 \\ A_p(2(1/(1+e^{-m_p 2x}))) - A_p & si\ x > 0 \end{cases} \tag{11}$$

where parameters A_n and A_p modify the saturation limits, whereas m_n and m_p modify the manner of the transition between those limits. Subscripts *p* and *n* refer to positive and negative values, respectively. Therefore, they are defined as:

$$g_1(I) = tgH(c_I(I)) \tag{12}$$

$$g_2(V) = tgH(c_V(V)) \tag{13}$$

$$g_3 = 0 \tag{14}$$

The previous functions do not consider all stages of the power plant and according to the input scenarios, the MPPT algorithms that are usually used could reduce the performance. Considering this situation, the penalty function g_3 is added. The reason for including this penalty and not adding the global output to the fitness function is to avoid a higher computational cost. In this way, only solutions that obtained the highest fitness values in the evolution are evaluated with the complete simulator. Two types of g_3 are proposed, the first one given by

$$g_3 = |E| \tag{15}$$

where $|E|$ is the cardinality of the subset of extremes E. The second one can be expressed as

$$g_3 = \left| E_u \right| \tag{16}$$

where $|E_u|$ is the number of pairs of adjacent extremes e_i that exceed the threshold u so that

$$\left| e_{i+1} - e_i \right| > u \tag{17}$$

Note: The extremes correspond to the maxima of the output curve (power-voltage). Two extremes are adjacent if are the closest on its coordinates of tension.

Fitness functions 1, 2, 3 and 4 with the addition of the first penalty g_3 are called functions 5, 6, 7 and 8. A similar situation arises for fitness functions 9, 10, 11 and 12 but using the second g_3 Out of expediency, from now on the said penalties will be called p_1 *and* p_2

3. RESULTS

This section presents the performance obtained through the application of the proposed methodology. First, experiments are carried out to determine a crossover rate and the best way to initiate the algorithm for the problem under study. Both determinations will be applied to the rest of the experiments in this chapter. Then, the results for different mutation rates are presented and discussed, using specially designed operators and the proposed variations of fitness functions.

In these experiments, the fitness function is not directly used as a measure of goodness of the solutions found. Instead, the efficiency of the system is used in order to obtain results that can be compared to the results of Sanchez Reinoso, Milone, and Buitrago (2013). However, the fitness function is employed to lead the search since it adds the problem constraints. The efficiency of the system is the result of consid- ering the relationship between the output power and the power that would be obtained from the system without considering the losses caused by interconnections and, therefore, with maximum power point trackers finding the maximum points of their respective curves effectively. Each module will present an individual instant efficiency before the input data. The method used in this chapter to calculate the efficiency of the system allows obtaining solutions that are independent of the above-mentioned instant efficiencies and they only depend on the configuration of the system. This is very useful in spite of not being independent of the material used in the production of modules. A corresponding simulator must be used for each kind of material and the evolutionary method will find the most adequate solution for each case.

Simulations employed consider seven cloudy days under the same scenario described in Sanchez Reinoso, Milone and Buitrago (2013). This allows maintaining the shading conditions of the said paper and, besides, an extra difficulty is added when the optimization horizon is expanded. In order to make weather conditions more real, real radiation and temperature data were used, which were measured at the weather station belonging to the Weather Research Center of the School of Engineering and Water Sciences located within the facilities of the University "Universidad Nacional del Litoral". Since the results of power plant generations are calculated normalizing by the nominal power, it is possible to gain independence of the size of the power plant. Therefore, arrays of 32 photovoltaic modules were used to avoid high computational costs, but maintaining, at the same time, the possibility of comparing the results obtained.

Initializations

Experiments were carried out with the different types of initialization: deterministic, random and hybrid. Some works indicate that a range of high crossover probability values and a low mutation probability value may be suitable (Grefenstette, 1986; Schaffer, Caruana, Eshelman, & Das, 1989). These tests took crossover probabilities between 0.7 and 0.9, whereas the mutation probability value was 0.02. The fitness
function employed for the sake of simplicity is f_1 and a population of 150 individuals was used to evolve for 500 generations. The selection process employed was *competition* and elitism was used to keep the best solution in each generation. A competition consists in choosing completely random of individuals (k> 1); these individuals compete through its fitness and the winner is selected (this paper uses k = 4). Elitism was implemented conserving the best individual to the next generation.

Table 1 shows the results of the experiments (30 searches) carried out with different crossover probabilities and the three types of initializations. The results found demonstrate that a better performance is obtained using a crossover probability of 0.8. On the other hand, the results obtained when using the most common initialization —random— are better than the ones obtained with the deterministic initialization. Nevertheless, a hybrid initialization is the one that gave the best results; therefore, it will be used in the following experiments, particularly with the above-mentioned crossover probability.

Operators and Fitness Functions

Experiments were carried out with the different types of mutation operators and fitness functions proposed to solve the problem under study. These experiments were made using mutation rates between 2 and 10%, and a c_p = 80%, that gave the best results in previous experiments. The population size was 150 individuals and optimization stopped after 600 generations. Other factors considered in the experiments are the twelve fitness functions and the three mutation operators proposed. Results of 10 searches for each combination of parameters were evaluated and they are shown in Tables 2, 3 and 4.

The lowest mutation probabilities are the ones which gave the best results, in particular the m_p= 0.02 (Table 2). Regarding the operators, the results obtained with f_1 and m_p of 0.02 and 0.04 do not show significant improvements when the operator used is changed, being the performance of the best solutions around 0.6. In the case of f_2, in general, a higher performance is observed when using the inversion and exchange mutation operators. However, the ungroup operator with its best solutions achieves a perfor- mance of 0.71, which is not too far from the results obtained with the other operators. With f_3, the best solutions obtained when using the ungroup and exchange operators show a higher performance than

Table 1. Efficiency for different initializations varying the crossover probability

c_p	Deterministic	Random	Deterministic-Random
0.7	0.55	0.59	0.70
0.8	0.60	0.65	0.72
0.9	0.53	0.67	0.69

Table 2. Efficiency achieved with fitness functions 1, 2, 3 and 4 with inversion, ungroup and exchange mutation operators

Efficiency		Inversion		Ungroup		Exchange	
	m_p	**μ**	**Better**	**μ**	**Better**	**μ**	**Better**
f_1	0.02	0.69	0.72	0.49	0.61	0.62	0.64
	0.04	0.63	0.65	0.55	0.63	0.62	0.63
	0.06	0.66	0.67	0.53	0.61	0.60	0.62
	0.08	0.53	0.55	0.45	0.54	0.52	0.58
	0.1	0.40	0.42	0.41	0.52	0.51	0.54
f_2	0.02	0.74	0.79	0.52	0.71	0.75	0.77
	0.04	0.75	0.75	0.53	0.64	0.74	0.76
	0.06	0.62	0.73	0.39	0.56	0.66	0.68
	0.08	0.53	0.60	0.35	0.44	0.63	0.64
	0.1	0.45	0.59	0.32	0.41	0.60	0.66
f_3	0.02	0.68	0.69	0.65	0.77	0.82	**0.84**
	0.04	0.65	0.69	0.72	0.75	0.77	0.82
	0.06	0.37	0.55	0.50	0.55	0.57	0.74
	0.08	0.22	0.41	0.32	0.51	0.49	0.62
	0.1	0.21	0.38	0.43	0.54	0.39	0.51
f_4	0.02	0.80	0.83	0.61	0.65	0.84	**0.86**
	0.04	0.79	0.81	0.71	0.72	0.82	0.836
	0.06	0.68	0.75	0.52	0.64	0.39	0.51
	0.08	0.61	0.74	0.33	0.42	0.35	0.53
	0.1	0.42	0.66	0.43	0.50	0.37	0.42

Table 3. Efficiency achieved with fitness functions 5, 6, 7 and 8 with the inversion, ungroup and exchange mutation operators

Efficiency		Inversion		Ungroup		Exchange	
	m_p	**μ**	**Better**	**μ**	**Better**	**μ**	**Better**
f_5	0.02	0.68	0.71	0.38	0.70	0.54	0.60
	0.04	0.77	0.78	0.7	0.75	0.76	0.79
	0.06	0.54	0.61	0.42	0.54	0.62	0.65
	0.08	0.41	0.57	0.43	0.53	0.59	0.63
	0.1	0.32	0.51	0.27	0.44	0.5	0.57
f_6	0.02	0.71	0.73	0.62	0.68	0.78	0.8
	0.04	0.74	0.75	0.74	0.79	0.81	0.82
	0.06	0.65	0.67	0.53	0.62	0.58	0.60
	0.08	0.42	0.50	0.44	0.48	0.54	0.57
	0.1	0.48	0.52	0.37	0.43	0.41	0.51
f_7	0.02	0.71	0.75	0.57	0.77	0.84	0.85
	0.04	0.63	0.77	0.71	0.81	0.85	**0.87**
	0.06	0.30	0.35	0.44	0.49	0.46	0.51
	0.08	0.22	0.43	0.24	0.40	0.42	0.47
	0.1	0.19	0.32	0.41	0.47	0.35	0.42
f_8	0.02	0.64	0.66	0.72	0.83	0.81	0.87
	0.04	0.72	0.79	0.8	0.84	0.85	**0.89**
	0.06	0.69	0.70	0.59	0.63	0.4	0.56
	0.08	0.66	0.68	0.19	0.48	0.36	0.47
	0.1	0.53	0.55	0.41	0.45	0.21	0.39

Table 4. Efficiency achieved with fitness functions 9, 10, 11 and 12 with the inversion, ungroup and exchange mutation operators

Efficiency		Inversion		Ungroup		Exchange	
	m_p	**μ**	**Better**	**μ**	**Better**	**μ**	**Better**
f_9	0.02	0.69	0.73	0.70	0.75	0.64	0.67
	0.04	0.66	0.70	0.67	0.72	0.64	0.65
	0.06	0.60	0.64	0.55	0.62	0.56	0.62
	0.08	0.54	0.57	0.54	0.59	0.51	0.60
	0.1	0.49	0.53	0.48	0.55	0.50	0.58
f_{10}	0.02	0.73	0.74	0.75	0.80	0.77	0.82
	0.04	0.70	0.79	0.72	0.77	0.72	0.79
	0.06	0.67	0.7	0.54	0.61	0.60	0.64
	0.08	0.60	0.63	0.51	0.54	0.53	0.55
	0.1	0.53	0.61	0.45	0.48	0.44	0.52
f_{11}	0.02	0.75	0.79	0.78	0.84	0.81	0.90
	0.04	0.64	0.70	0.74	0.84	0.81	0.89
	0.06	0.57	0.60	0.59	0.63	0.69	0.76
	0.08	0.51	0.55	0.53	0.58	0.57	0.60
	0.1	0.46	0.52	0.49	0.57	0.53	0.57
f_{12}	0.02	0.74	0.8	0.76	0.81	0.82	**0.92**
	0.04	0.75	0.83	0.68	0.79	0.80	**0.90**
	0.06	0.70	0.74	0.61	0.65	0.56	0.59
	0.08	0.64	0.68	0.46	0.51	0.47	0.52
	0.1	0.53	0.59	0.29	0.32	0.46	0.50

the one achieved when using f_1 and f_2 with the said operators. The reason may be a better design of the fitness function that penalizes more strongly at the extreme of the allowable input range of the inverters. However, with f_3 and the inversion operator, the best performance values are lower than the one found using f_2 and the same operator. Probably, this is caused by higher exploration requirements, whereas the operator in question is less disruptive than the rest. When using f_4, the inversion and exchange operators show a higher performance than for the other fitness functions cases, whereas the best results obtained by the ungroup operator are slightly lower than when f_3 is used. The results obtained with f_4 would suggest that including penalties with saturation is beneficial, thus reducing the number of individuals that are excessively penalized and that affect the convergence of the population. An analysis of the complete table indicates that the best results are obtained with the exchange operator, particularly for cases in which f_3 and f_4 are used. With f_3 a performance of 0.84 is achieved and with f_4 the value reaches 0.86. It is known that many maximum power point tracking algorithms are based on first-order search methods and that those methods can undergo stagnation in local minimums. A possible cause for not obtaining higher performance values may be that an optimum array can give out the maximum global power but it may be difficult for the maximum power point tracking algorithm to explore the morphology of the output curve.

When f_7 and f_8 are used with m_p = 0.06, 0.08 and 0.1, the best performance values obtained are lower than using the same probabilities with f_3 and f_4 (Table 3). Using identical mutation probabilities, the inversion operator obtains higher performance values with f_5 than when it is used with f_1. Nevertheless, when using it with f_6, the best performance values are lower than when using f_2. If the results obtained with f_5 and f_6 are compared to the results obtained with f_1 and f_2 using the ungroup operator, lower performance values can be observed. However, with the exchange operator, the best performance values increase, all this with the probabilities mentioned previously. Table 3 shows that the highest performance is achieved with $m_p = 0.04$, in contrast with when $f_{1\text{-}4}$ are used, which show a higher optimality for $m_p = 0.02$. This would be caused by the influence of the g_3 penalty that reflects the output multimodality and would require a higher mutation to find the global optimum. The best results using $f_{5\text{-}8}$ with the designed operators show better solutions for the exchange operator that, in fact, are better than when using $f_{1\text{-}4}$. Under such situations, attacking the problems of the MPPT in cases of shading through the g_3 penalty was favorable. In fact, it is worth mentioning that this was achieved without modifying the tracking algorithm per se. Improving the algorithm would increase the global performance even more.

Table 4 shows the results obtained when using functions $f_{9\text{-}12}$ and the best performance values are, in general, higher than the ones obtained with the other fitness functions. Out of expediency, from now on g_3 penalties of those functions will be called p_2, and g_3 penalties used in $f_{5\text{-}8}$ will be p_1. Using p_2 penalty yielded better results than using p_1. In the case of n peaks of equal magnitude, this could be explained because p_1 would be equal to n, whereas p_2 would be equal to 0. In such cases, the correct penalty should be equal to 0 since the aim is to optimize power; therefore, only the voltage magnitude would vary but that magnitude is already considered by the corresponding constraint. Consequently, p_1 could be considered as a maximum value of p_2. Although, in general, better results can be obtained with p_2 than with p_1, there are few cases in which functions that do not include a peak penalty achieve a better performance. These cases may be due to the fact that the maximum power point tracking works especially well under that scenario and/or the thresholding penalty led to the rejection of solutions that could have been optimum enough at the moment of evolving. When selecting the threshold for p_2, relative magnitude differences higher than 0.35 were considered for counting the peaks. Using this value is a conservative strategy since we do not want to penalize too strongly solutions that could be promising at the time of evolving. Besides, adopting differences of very small magnitude makes us tend to the case of using p_1, and it is clear that if the difference among power peaks is small, preference of one extreme over others must not exist. Considering the fitness functions and operators used in the experiments, the best performance values are achieved by f_{12} function and the exchange operator, and the value is around 0.9.

For greater clarity results are synthesized for different fitness functions and operators in Figure 12. In addition, is shown the change fitness of the best individual in depending on number of generations (Figure 13).

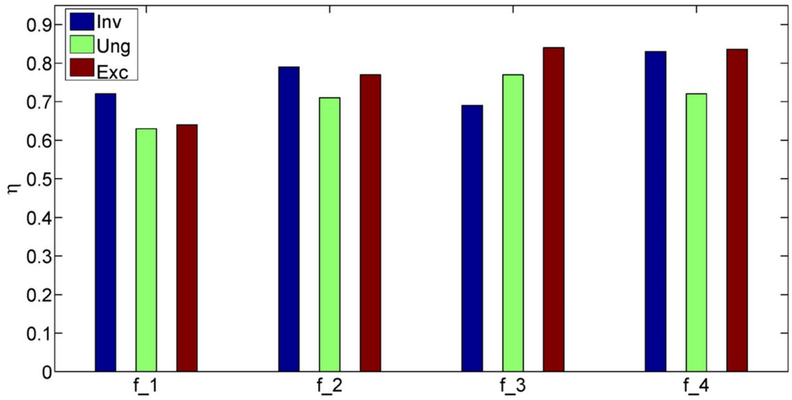
Results show a promising performance if compared to the design of some photovoltaic power plants that are currently installed and that were monitored for several years, showing a minimum average efficiency of 56% and a maximum average efficiency of 71%.

Evolved Configurations

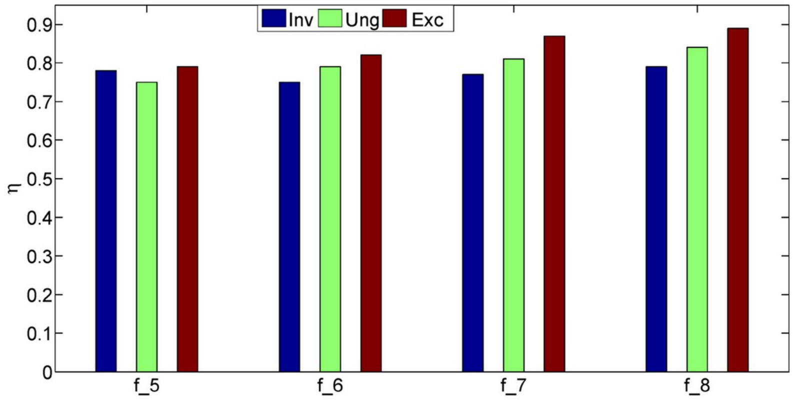
Examples of evolved configurations will be now presented. The results presented were subjected to an (automatic) simplification process. The simplification consists of removing one root and connecting all its descendants to the parent of the said root. The purpose is to eliminate the redundancy in the tree-based representation, thus allowing a better interpretation of its meaning from a physical point of view. This operation does not modify the series-parallel relationship of its elements, which means that they represent the same connection and, therefore, the final output of the array is not altered.

Figure 14 exemplifies simplification cases from three configurations. Figure 7(a), 7(b) and 7(c) show trees before the simplification operation is applied. It should be observed that after applying the operator

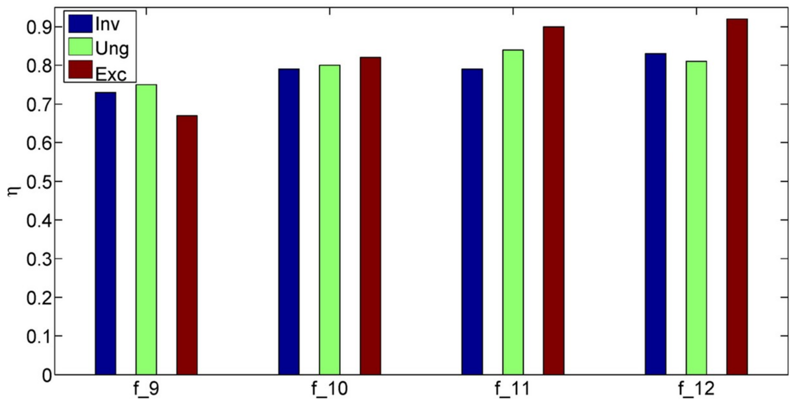
Figure 12. Best global efficiencies for different fitness functions and operators. In Figure 12(a) are shown the results for the functions 1-4, in Figure 12(b) for the functions 5-8 and in Figure 12(c) for the functions 9-12



(a)



(b)



(c)

Figure 13. Efficiency as a function of the number of generations

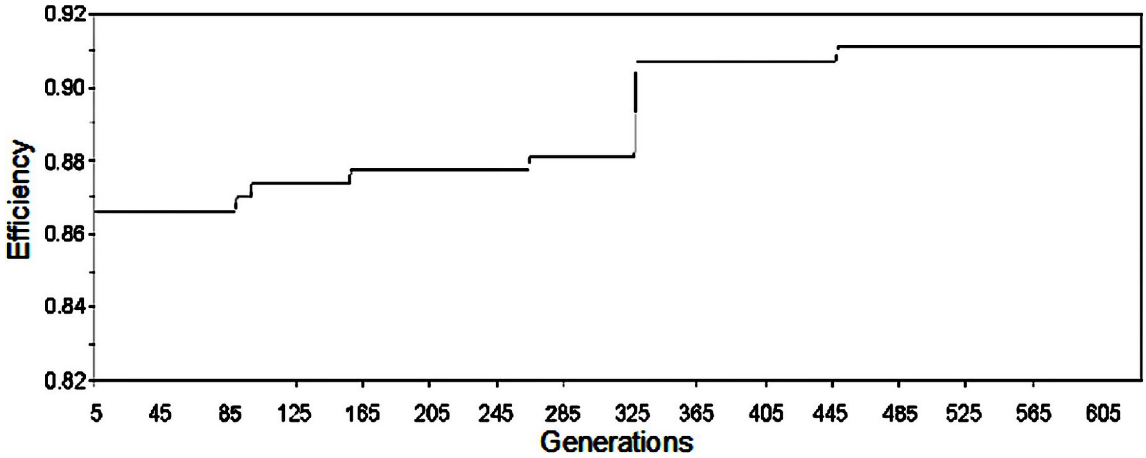
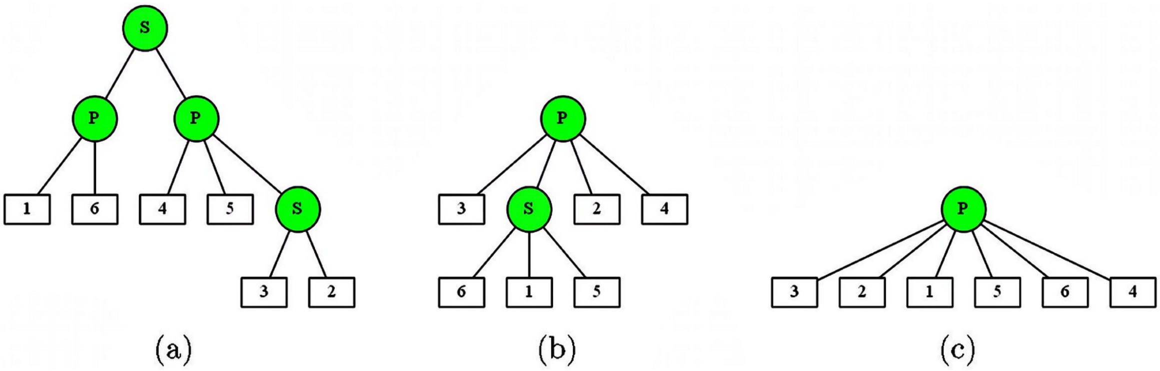


Figure 14. Simplified configurations of Figure 7(a), 7(b) and 7(c), respectively



to those configurations, it is possible to obtain representations with a reduction of 16.6%, 25% and 80%, respectively, of the number of nodes required to represent the same connection (Figure 14(a), 14(b), 14(c)). The best results in each table have certain similarity (Figure 16). This could be attributed to the fact that they are obtained using the same fitness function and operator. Table 2 shows the results with the highest similarity among themselves and Table 4 shows the ones having the lowest similarity. This could be explained by the different mutation probabilities that produced those results. Regarding Table 3, the best configurations are relatively more complex than the ones in the other tables. The common characteristic of the best solutions in all tables is that they show a maximum use of allowable current and a lower use of voltage. In addition, the current-voltage use relationship among the solutions is similar. In order to analyze the common aspects in the solutions, the complexity of the configuration will be used as a similarity measure and this complexity will be evaluated according to the number of groups by level. Therefore, a complexity by level and a global complexity given bythe sum of complexities will be considered.

Figure 15. Complexity of solutions (C): Configuration (circuit) with four groups of level one (C_1=4), two groups of level two (C_2=2). The global complexity is calculated as C_1+C_2

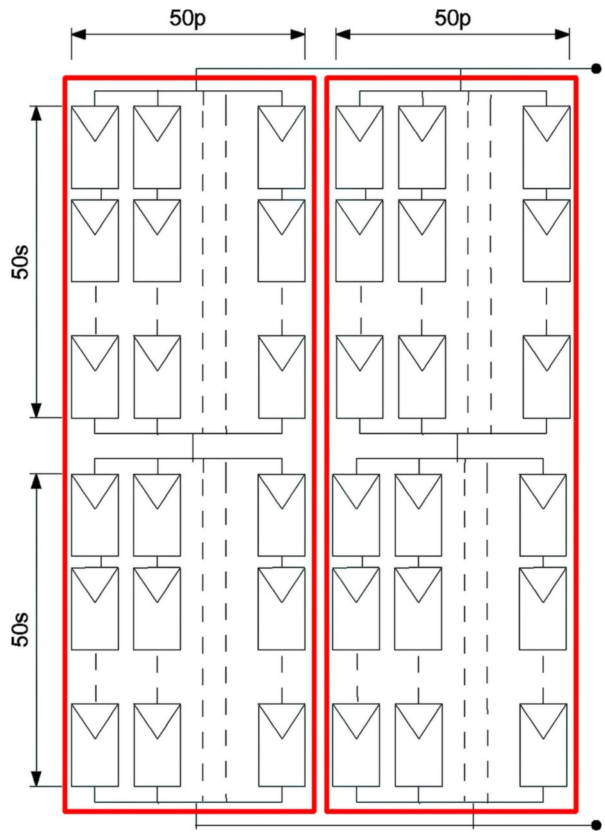
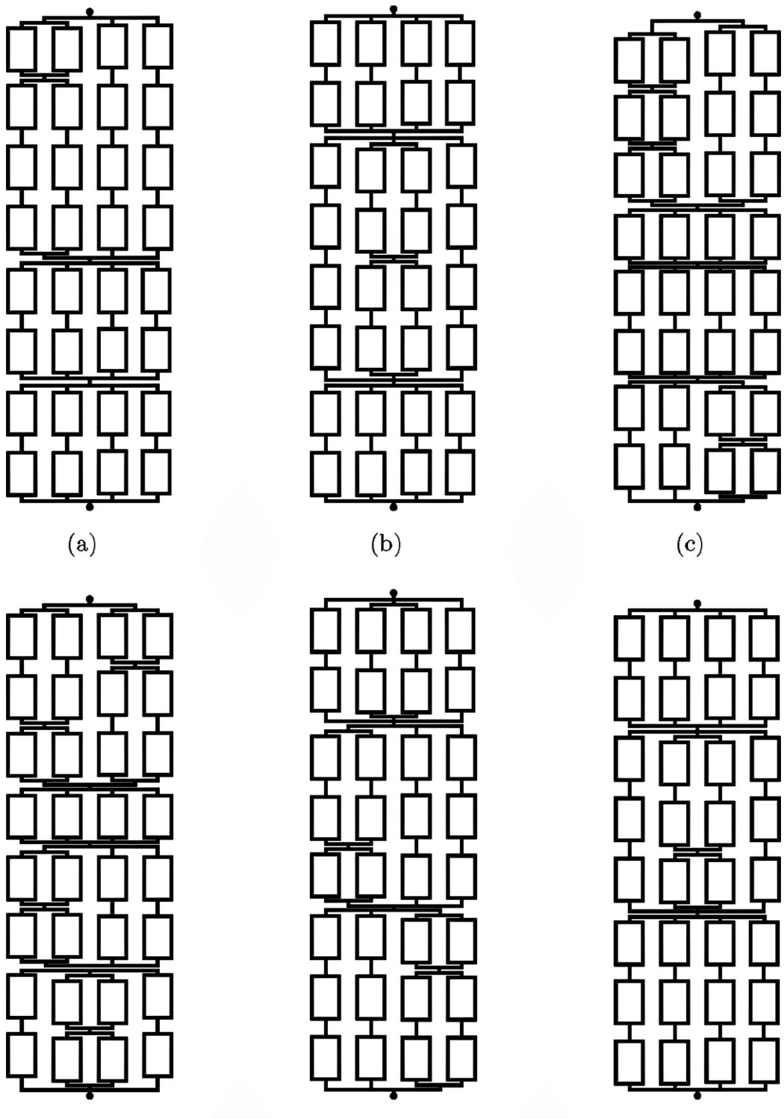


Figure 16. Evolved configurations. a) and b) show the best configurations in Table 2, c) and d) show the best configurations in Table 3, and e) and f) show the ones in Table 4



The level indicates the order of precedence which is employed to solve the circuit. The said circuit consists of groups (sub-circuits) in each level. To give one example, the configuration shown in Figure 15 indicates four groups of level one consisting of 50 strings in parallel (50p) with 50 modules in series (50s) for each string, and that there are two groups of level two in parallel, which consist of level one groups connected in series.

In the cases of Figure 16(a) and 16(b), the complexity of the highest and lowest level of these configurations coincides and so does their global complexity. Considering the number of highest level groups, configurations of Figure 16(c) and 16(d) are similar and the number of lowest level groups is equal. If the lowest level groups are observed, they also have some common characteristics. However, the solutions are more complex that those obtained in the other tables. This is reflected in the higher total number of groups considering all levels. On the other hand, within the best configurations obtained in Table 4 (Figure 16(e) and 16(f)), the lowest level similarity is the same but the highest level similarity differs. This difference is higher among the pairs of best configurations found. This could be due to the different mutation probabilities with which they were obtained.

In general terms, the solutions do not have many nesting levels. Possibly, using a higher number of modules may provide solutions of a higher number of levels. In the experiments carried out, the proliferation of levels was strongly limited by voltage and current constraints. A similar effect can be caused by the fact that the number of modules must be constant. Also, it is necessary to consider that the redundant levels of the evolved solutions are eliminated by the simplification process. Nevertheless, it is worth explaining that, in practice, it is desirable to obtain a low number of connection levels in the systems. In addition, the representations obtained with this method, that is, the best chromosomes, may have a structure that is not simple to analyze. Future works will evaluate the possibility of obtaining general conclusions about the structure of the obtained representation and about the corresponding electric systems.

4. CONCLUSION

The objective of this chapter was to optimize the photovoltaic power plant considering the effects of variable shading on time and weather. For that purpose, an optimization scheme based on a simulator and on genetic programming techniques were proposed.

The effect of crossover probability and initialization on the performance measures proposed for the analysis was studied and it was observed that the best results are obtained with 80% rates. The proposed hybrid initialization makes possible to obtain higher performance values than when the deterministic or random initialization are used exclusively. Also, the results obtained for different mutation rates were analyzed when the three designed mutation operators and the four proposed fitness functions are used based on the performance of the system. Regarding the mutation rate, in general, the best performance is achieved with a m_p = 0.02 in most cases and to a lesser extent with a m_p = 0.04. The proposed exchange operator is the one that obtains higher performance values, particularly with fitness functions with exponential penalties and hyperbolic tangent. The said performance values reach 0.92 and 0.90, respectively, turning them into promising results considering the performance currently obtained in operating photovoltaic power plants.

The modeling methodology and the strategy to implement it provide a useful engineering tool to analyze the performance of a photovoltaic system, to evaluate and design connection schemes, reducing the probability of poor quality designs and high costs, and improving the power productivity and the efficiency of the system.

REFERENCES

Chao, K., Chiu, C., Li, C., & Chang, Y. (2011). A novel neural network with simple learning algorithm for islanding phenomenon detection of photovoltaic systems. *Expert Systems with Applications*, *38*(10), 12107–12115. doi:10.1016/j.eswa.2011.02.175

Díaz-Dorado, E., Suárez-García, A., Carrillo, C. J., & Cidrás, J. (2011). Optimal distribution for photo-voltaic solar trackers to minimize power losses caused by shadows. *Renewable Energy*, *36*(6), 1826–1835. doi:10.1016/j.renene.2010.12.002

Enrique, J., & Durán, E. (2010). A reliable, fast and low cost maximum power point tracker for photo-voltaic applications. *Solar Energy*, *89*(1), 79–89. doi:10.1016/j.solener.2009.10.011

Esram, T., & Chapman, P. (2007). Comparison of photovoltaic array maximum power point tracking techniques. *IEEE Transactions on Energy Conversion*, *22*(2), 439–449. doi:10.1109/TEC.2006.874230

Gómez-Lorente, D., Triguero, I., Gil, C., & Espín Estrella, A. (2012). Evolutionary algorithms for the design of grid-connected PV-systems. *Expert Systems with Applications*, *39*(9), 8086–8094. doi:10.1016/j. eswa.2012.01.159

Grefenstette, J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *16*(1), 122–128. doi:10.1109/TSMC.1986.289288

Houssamo, I., Locment, F., & Sechilariu, M. (2010). Maximum power tracking for photovoltaic power system: Development and experimental comparison of two algorithms. *Renewable Energy*, *35*(10), 2381–2387. doi:10.1016/j.renene.2010.04.006

Koca, A., Oztop, H. F., Varol, Y., & Koca, G. O. (2011). Estimation of solar radiation using artificial neural networks with different input parameters for mediterranean region of anatolia in turkey. *Expert Systems with Applications*, *38*(7), 8756–8762. doi:10.1016/j.eswa.2011.01.085

Kornelakis, A., & Marinakis, Y. (2010). Contribution for optimal sizing of grid-connected PV-systems using PSO. *Renewable Energy*, *35*(6), 1333–1341. doi:10.1016/j.renene.2009.10.014

Larbes, C., Ait Cheikh, S., Obeidi, T., & Zerguerras, A. (2009). Genetic algorithms optimized fuzzy logic control for the maximum power point tracking in photovoltaic system. *Renewable Energy*, *34*(10), 2093–2100. doi:10.1016/j.renene.2009.01.006

Martínez-Moreno, F., Muñoz, J., & Lorenzo, E. (2010). Experimental model to estimate shading losses on PV arrays. *Solar Energy Materials and Solar Cells*, *94*(12), 2298–2303. doi:10.1016/j.solmat.2010.07.029

Mellit, A., Kalogirou, S. A., & Drif, M. (2010a). Application of neural networks and genetic algorithms for sizing of photovoltaic systems. *Renewable Energy*, *35*(12), 2881–2893. doi:10.1016/j.renene.2010.04.017

Mellit, A., Mekki, H., Messai, A., & Kalogirou, S. (2011). FPGA-based implementation of intelligent predictor for global solar irradiation, part i: Theory and simulation. *Expert Systems with Applications*, *38*(3), 2668–2685. doi:10.1016/j.eswa.2010.08.057

Mellit, A., Mekki, H., Messai, A., & Salhi, H. (2010b). FPGA-based implementation of an intelligent simulator for stand-alone photovoltaic system. *Expert Systems with Applications*, *37*(8), 6036–6051. doi:10.1016/j.eswa.2010.02.123

Messai, A., Mellit, A., Guessoum, A., & Kalogirou, S. (2011). Maximum power point tracking using a GA optimized fuzzy logic controller and its FPGA implementation. *Solar Energy*, *85*(2), 265–277. doi:10.1016/j.solener.2010.12.004

Moradi, M., & Reisi, A. (2011). A hybrid maximum power tracking method for photovoltaic systems. *Solar Energy*, *85*(11), 2965–2976. doi:10.1016/j.solener.2011.08.036

Ozgoren, M., Bilgili, M., & Sahin, B. (2012). Estimation of global solar radiation using ANN over turkey.
Expert Systems with Applications, *39*(5), 5043–5051. doi:10.1016/j.eswa.2011.11.036

Petrone, G., Spanuolo, G., & Vitelli, M. (2007). Analytical model mismatched photovoltaic fields by means of lambert w-function. *Solar Energy*, *91*(18), 1652–1657.

Sanchez Reinoso, C. R., Milone, D. H., & Buitrago, R. H. (2013). Simulation of photovoltaic centrals with dynamic shading. *Applied Energy*, *103*, 278–289. doi:10.1016/j.apenergy.2012.09.040

Schaffer, D., Caruana, A., Eshelman, L., & Das, R. (1989). Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In *Proceedings of Third International Conference on Genetic Algorithms* (vol. 1, pp. 51-60). Academic Press.

Sivanandam, S., & Depa, S. (2008). *Introduction to Genetic Algorithms*. Berlín: Springer.

Sullivan, C., Awerbuch, J., & Latham, A. (2011). Decrease in photovoltaic power output from ripple: Simple general calculation and effect of partial shading. In *Proceedings of Twenty-Sixth Annual IEEE Applied Power Electronics Conference* (vol. 2, pp. 1954-1961). IEEE. doi:10.1109/APEC.2011.5744864

Ubisse, A., & Sebitosi, A. (2009). A new topology to mitigate the effect of shading for small photovoltaic installations in rural sub-saharan africa. *Energy Conversion and Management*, *50*(7), 1797–1801. doi:10.1016/j.enconman.2009.03.016

KEY TERMS AND DEFINITIONS

Crossover: Operation used in evolutionary algorithms to model the genetic crossover that occurs in the natural evolution.

Evolved Configuration: Configuration obtained by applying the process of natural evolution in a certain problem.

Initialization: Method for setting initial conditions enabling to an algorithm to perform the calculations. It is desirable to properly design the method to get the best possible performance of the algorithm.

Mutation: Operation used in evolutionary algorithms to model the genetic mutation that occurs in the natural evolution.

Optimization: Theory that under certain criteria allows to minimize or maximize functions and satisfy limitations.

Photovoltaic System: System for generating electricity from solar energy and semiconductors. This mainly consists of photovoltaic modules and the power conversion stage which decomposes into a DC and an AC stage.

Set of Solutions: A set whose elements are evolved configurations.