Multiagent System powered by Neural Network for positioning control of solar panels

An optimization for sun tracking systems

D. Oviedo, M.C. Romero-Ternero, A. Carrasco, F. Sivianes, M.D. Hernandez, J.I. Escudero
Electronic Technology Department
University of Seville
ETSI Informatica, Avda. Reina Mercedes s/n 41012 Seville (Spain)
oviedo@dte.us.es, mcromerot@us.es, acarrasco@us.es, fsivianes@us.es, marilohv@dte.us.es, ignacio@us.es

Abstract—**This paper presents a model of neural network for position control of solar panels in multiagent-based control systems. This neural network is integrated within agents in order to optimize and predict the best positioning of solar panels depending on the position of the sun and other variables. The agents in this system can cooperate and coordinate to achieve a sun tracking system optimized, simple and adaptive.**

Keywords—control systems; solar energy; position control; renewable energy; multi-agent system; sun tracking; neural network

I. Introduction

In order to improve the efficiency of power generation in solar photovoltaic energy plants, it is required to begin with improving the tracking accuracy of solar panels. Two main elements are necessary to do this: a distributed system to obtain environmental information and an adaptive sun tracking system.

Hardware diversity of sensors and actuators required and its location in a solar power plant make advanced control systems necessary. A distributed intelligent agent architecture provides flexible and scalable ways to integrate different sensors and actuators.

In addition to the multi-agent system, we propose a neural network model that is able to predict the optimal movement patterns, taking all information collected by the multi-agent system and the behaviors and results obtained earlier in time.

Our neural network presents an adaptive two-dimensional auto-tracking control method suitable for different weather conditions that uses real-time control strategies and control parameter auto-optimization methods to realize flexible timing tracking and photoelectric tracking control.

This model not only implies some guidelines on how the neural network can be designed, but also defines the integration of the elements of our system. The proposed model is applied to a control system of a solar power plant, obtaining an architecture which enables system agents to solve the problem. The agents in this system can cooperate and coordinate to achieve the global goal of optimization in energy generation.

This paper is organized as follows:

Section 2 presents problem domain to be solved to improve the energy efficiency of solar panels, while section 3 details the basic concepts of the model and the proposed neural network. In section 4 integration of the neural network with multi-agent system is described and the concluding remarks are found in section 5.

II. Problem Domain

In solar photovoltaic panel, trackers are used to minimize the angle of incidence between the sun insolation and a photovoltaic panel [1]. This increases the amount of energy produced compared to fixed panels as shown in Figure 1. This may result in an increase of 50% of the energy efficiency, depending on the environment.

Unsuitable environments (shades, mountains, weather conditions, etc.) may result in decreases for energy efficiency achieved with traditional trackers [2][3].

Generally, the solar tracking controllers are microcontroller-based system which has been designed to follow up on two axes to with high precision. The control modules works based on the theoretical coordinates of the position of the sun, without considering details of the

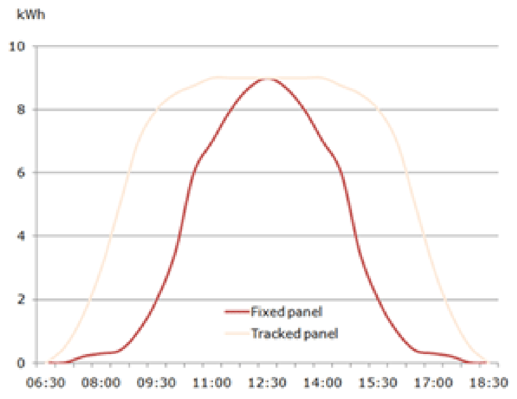


Fig. 1. Difference in output energy between fixed panels and tracked panels

environment. In some cases, the systems have a solar sensor that works when there is not enough radiation directly and includes an auto adjustable function for positioning the solar panel.

In any case, these adjustments are not very precise because it does not consider all variables that can affect to the global radiation produced by the sun. Some elements that can affect to the performance are dynamic, as shades or weather conditions. Therefore, an adaptive system that considers these dynamic elements and previous behaviours may improve performance.

In order to create a solar adaptive tracking system based in real-time data from environment, devices to obtain environmental information and a mechanism to predict movements according to information obtained are necessary. Data acquisition can be implemented through a multi-agent system with sufficient capacity to treat all information adequately [4].

To achieve an adaptive positioning system, a good solution is a neural network problem. A particular type, called recurrent neural networks [5], consists of neural networks that include parameters defined through the time, allowing these networks have a dynamic behaviour or temporarily. This feature means that they are able to process time sequences because it can remember the relevant history of the sequences partially. Consequently, these networks are characterized to allow prediction of events or significant patterns because it can propagate the information stored in the network to forward in time [6].

III. NEURAL NETWORK MODEL FOR SOLAR TRACKING

The traditional trackers are based on a theoretical determination of the orientation, tilt and angle of incidence in a solar panel for maximum theoretical performance [7] and it is possible that these calculations do not correspond to the final circumstances. Furthermore, these methods do not consider diffuse radiation and reflected radiation separately.

There are environments where movement patterns may not be optimal. Examples such as special places like the forest, fog areas, interior areas and places with complicated shape areas can be found. Generally, it is advisable to evaluate the performance during long periods of time, such as one year.

Fixing the movements of the solar panel from a previous theoretical study may lead to performance losses. We propose use a neural network for adapting positioning moves of solar panel, considering real environment. Our model considers position with respect to the objects in the environment that can affect performance and changes about states theoretically expected (clouds, shadows, seasonal variations, etc).

Our neural network model fixes proper movement pattern by learning for a given day of a year, considering climate state, time schedule and solar radiation expected.

A. Scenario for modeling

The modeling scenario has a set of hardware devices for typical solar energy systems. This system has a two-axis solar tracking system in a solar array. With this system it is possible to track the sun, allowing solar radiation impinges perpendicularly on the solar panel due to movement generated for robotic two-axis mechanism: east-west azimuth axis (orientation) and top-down zenith axis (tilt).

The available system (SunTracking [8]) consists of a complete set of linear actuators that allows to move panels in two axes. It also provides a full hardware and software solution that allows to control any motors for solar tracking device, by two encoders in azimuth and zenith axes.

As sensors, there is a pyranometer (also called solarimeter or actinometer) which is a sensor designed to measure the solar radiation flux density from a field of view of 180 degrees. Other available sensors and devices can provide us information as Solar Time UTC, azimuth and zenith sun coordinates, day of year, temperature, humidity, wind speed, atmospheric pressure and weather conditions (sunny, cloudy, rain, etc).

Finally, it is also possible to obtain a measurement of system performance in KWh.

B. Type of neural network

The model developed is a neural network that can determine an appropriate movement pattern to maximize performance of general system in a particular time of day for a concrete year, in the environment in which a solar panel is located.

According to problem characterization, it is appropriate to use a recurrent neural network fully connected type. This type of network is defined by dynamic behaviour. It allows temporal information processing, such as dynamic patterns depending on time and consideration of values at previous instants of time.

This network considers the previous position of the solar panel regarding previous instants and therefore it predicts the next coordinates of movement (which are close). In addition, training and output feedback is analyzed by comparing energy performance. It adjusts to the movements or output patterns that are most suitable.

To define network correctly, it is necessary to set the input and output vectors of the network.

C. Input Vector Network

Inputs network are values that should consider our neural network to determine subsequently output patterns. It is determined by domain problem to be solved. Our design includes all sensor values available, as well as positioning data of solar panel at instant "t" given by "Solar UTC Time":

- Azimuth and Zenith solar panel coordinates
- Solar time UTC
- Azimuth and Zenith sun coordinates
- Day and year
- Solar radiation (Kw/m2)
- Temperature

- Humidity
- Winds speed
- Atmospheric Pressure
- Weather conditions (sunny,cloudy, rain, etc)
- Energy performance (kWh)

Each of these elements can be encoded into binary level (the set of values that may have) or continuous 0-1. Then the number of input neurons is determined. The number of elements of the input vector is equal to the total number of neurons required by the values shown.

D. Output Vector Network

As the input vector, it depends on the intended result with the neural network. In the case of the proposed model, the coordinates will be sought on the zenith and azimuth angles to be targeted solar panel to maximize the time "t+1" given by "solar time UTC + Δt", where Δt will be the interval that desired define for testing a new movement.

The output neurons can be defined with binary values as the input vector.

E. Middle layers / Hidden Layers

There are no theoretical reasons for determining both the number of middle layers of the network, as the number of neurons must have [9].

On a practical level, a maximum of two middle layers is recommended in studies on neural networks [9], because there are no studies indicating the need to use more layers. Most problems can be solved with a single layer. General reasons for using the lowest number of layers are:

- Training slows when more hidden layers are used.
- The more number of layers, the more unstable is the error gradient.
- The more number of layers, the more number of local minimum it is obtained.

Our design starts using a single layer. In the case we observe that the time required in training or in responses is very high, and the results are not right, then the network is increased up to two hidden layers reducing the number of neurons in each layer.

F. Number of neurons in the hidden layer

When choosing the number of neurons in the hidden layer composed, we have considered that an excessive number of neurons may result in overadjustment (also known as overfitting) which is the effect of overtraining the network. This makes the learning algorithm be tailored to very specific characteristics of the training data that have no causal relationship with the intended results. On the other hand, a very small number of neurons can lead to the problem is not properly turned over the network.

There are no rules or algorithms that specify theoretically and demonstrated the number of neurons that the hidden layers should make up. However, there are valid practical rules due to its empirical verification.

Considering the details mentioned, there is a rule called pyramidal geometrical rule [10] that allows the number of neurons in the hidden layers to be approximated. This will distribute pyramidal neurons in decreasing order from the input neurons to the output ones. For a three-layer network with one hidden layer, this rule is defined by:

$$h = \sqrt{m * n} \qquad (1)$$

Where n is the number of input neurons, m the number of output neurons and h the initial number of neurons in the hidden layer place.

For a network with two hidden layers, the calculation of the number of neurons for the middle layers is given by the same rule, according to the following formulation:

$$h_1 = m * r^2 \qquad (2)$$

$$h_2 = m * r \qquad (3)$$

$$r = \sqrt[3]{\frac{n}{m}} \qquad (4)$$

Where h1 and h2 are the number of neurons for the first and second hidden layer respectively.

There are other types of rules for treating number of the hidden layers of neurons, taking into account the number of searched patterns or the number of neurons of the input layer [10]. Because these rules are of approximate nature, it was decided to use the one described above due to its simplicity, and combined with an iterative construction method for network growth.

G. Initialization of the weights

Due to the high number of input connections we decided to set a low value of weights, following as an approximate rule the fraction of the number of total weights to fix.

H. Learning of the network

The neural network must be trained with a sufficiently large data set, representative of the problem. In the case of the raised problem, a previously trained neural network did not exist because we do not know the expected output. The network must adapt to the outputs (new positioning of the panel) with better energy efficiency.

During training, the input to the neural network is a representation of the current position (along with the parameters of the environment) and the desired output is the representation of the next position. By having available an output to compare (energy efficiency), the learning algorithms can be supervised type.

To train the network under supervision normally required some measure of error e[t] to describe the adequacy of the output provided by the network to the desired value by adjusting the parameters of the network while trying to minimize this error. The neural network model has been designed using the quadratic error function [11], defined for time "t" as:

$$E[t] = \frac{1}{2}\sum_{i=1}^{nY}(d_i[t] - y_i[t])^2 \tag{5}$$

Where $d_i[t]$ is the desired output or target for the i-th output neuron at time t and $y_i[t]$ is the corresponding output of the network.

Due to the type of network chosen for modeling, the learning algorithm used is the backpropagation algorithm. This algorithm defines the network, consisting of the backward error propagation, i.e. the output layer towards the input one, via the middle hidden layers and adjusting the weights of the connections in order to reduce such error.

This learning stage is to obtain the minimum difference between the output obtained by the network and the output desired by the user. Therefore, this will be supervised training, as the user (supervisor) determines the desired output before presentation of a given input pattern.

I. Stop condition of network learning

A stop condition for training is defined in order to avoid network overtraining. This occurs when energy performance of the system output exceeding the expected theoretical performance.

To prevent erroneous training, due to temporary conditions or unanticipated circumstances, it has been defined the limit training as a maximum of 100 cycles, in case outputs do not improve the network performance with respect to the one initially established. In these cases, the expert system will have stored the initial positioning status, which will return the panel.

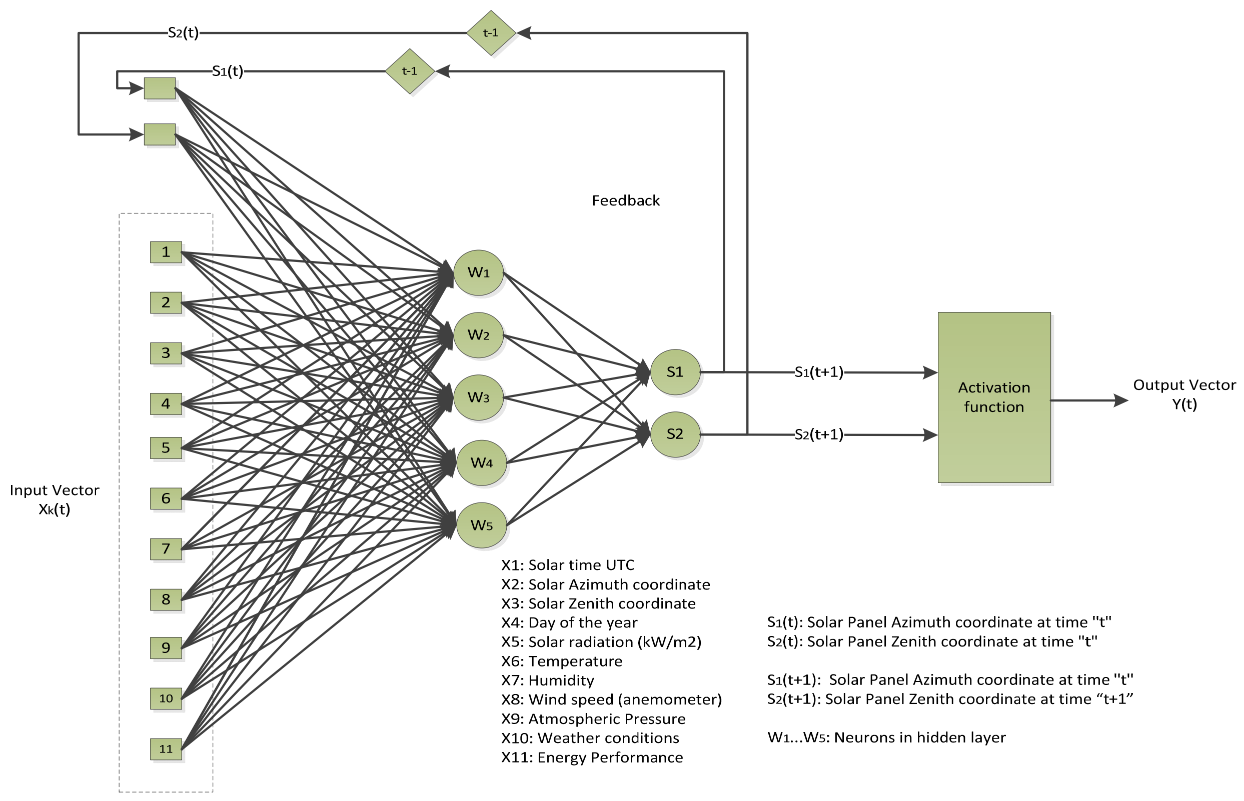
J. Neural Network Model

As indicated above, a recurrent neural network as network topology is selected. Within this a recurrent neural network completely on has been chosen, since there is no restriction of connectivity and all the input data are time dependent. In the same way, each neuron receives as input the activation of other neurons and its own activation.

$$a_i(t) = f_i\Big(\sum_{j\in A\cup B} w_{ji}a_j(t-1)\Big) \tag{6}$$

Where "A" represents the input neurons, "B" the rest of neurons and w_{ji} are the weights of the connections "j" to "i" between neurons.

As seen in Figure 2, the neural network reads a sequence of time and then the network outputs a possible continuation of the input sequence. In our case, only two future values are predicted, relative to the coordinates of the solar panel positioning as noted, the architecture consists of a network



S2(t)
S1(t)
t-1
t-1
Feedback
1
2
3
4
5
6
7
8
9
10
11
Input Vector
Xk(t)
W1
W2
W3
W4
W5
S1
S2
S1(t+1)
S2(t+1)
Activation function
Output Vector
Y(t)
X1: Solar time UTC
X2: Solar Azimuth coordinate
X3: Solar Zenith coordinate
X4: Day of the year
X5: Solar radiation (kW/m2)
X6: Temperature
X7: Humidity
X8: Wind speed (anemometer)
X9: Atmospheric Pressure
X10: Weather conditions
X11: Energy Performance
S1(t): Solar Panel Azimuth coordinate at time "t"
S2(t): Solar Panel Zenith coordinate at time "t"
S1(t+1): Solar Panel Azimuth coordinate at time "t"
S2(t+1): Solar Panel Zenith coordinate at time "t+1"
W1...W5: Neurons in hidden layer

Fig. 2. Neural Network Model for prediction of movements of solar panel in CARISMA project.

with an input layer and hidden layer processing elements totally interconnected, whose outputs are connected to the outputs of the network. The feedback is performed by interconnecting the output at time “t+1” to the corresponding inputs by delay elements.

It has been decided in this theoretical model temporarily defined and will adjust after empirical training:

- An input layer consists of two temporary neurons that serve as temporary input to the two current coordinates positioning of the solar panel and eleven general input neurons that make up the environmental status of the solar panel to the time “t”.
- A hidden layer consisting of five neurons, from the application of approximate rules previously seen. In any case, the number of neurons in the hidden layer may vary according to implementation needs.
- The output layer is composed by only two neurons, corresponding to the next coordinates of movement (coordinates of movement in azimuth and zenith “t+1”). These outputs are connected by feedback to find the temporary input neurons.

IV. Application of Model in Agents

The neural network model described is integrated in the CARISMA system [12]. CARISMA presents a multiagent architecture to obtain an integrated system to supervise solar farm infrastructures. This system is able to monitor the environment by using distributed sensors and collect the data from them. It provides an efficient maintenance and prevention in solar farms in order to act over the environment or to do recommendation automatically to the telecontrol operators.

The main actions of the agents are modeled by expert systems [4], using facts and rules. These rules represent real actions on devices such as cool systems, supply power or control position of a solar panel. Also these rules in an agent may represent the action required to share knowledge that may affect the positioning of the panels with other agents and reporting recommendations, faults or alarms in the system.

In order to provide software agents with the ability of identifying behaviour patterns that optimize the performance of the solar panels controlled by the system, it is necessary to integrate neural network modeled in the system. To achieve this inference, the control agents of the system implement a neural network that allows them to see patterns of behaviour that improve and optimize the overall system capacity to obtain solar energy. When the neural network find patterns that improve behaviour, the agent communicate it to the expert system already implemented in CARISMA. The expert system will be responsible for carrying out actions that lead to optimal states to the system, such as shown in the figure 3.

The expert system will provide the necessary information to the neural network and serves as a learning element for network through feedback. In any case, the expert system is responsible for the final decisions made in the system. The neural network will have an adaptive control and it will seek the best adaptation of the system to optimize performance over time.

The neural network will calculate the optimal movement patterns for solar panels to maximize energy production, through the analysis of the responses of the system for small perturbations generated by the expert system in such patterns movement.

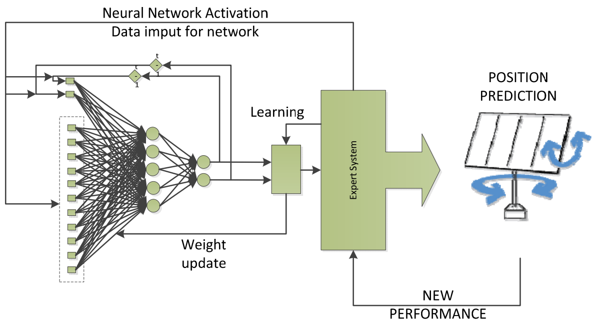


Fig. 3. Final model application in CARISMA project

Figure 4 shows the final network model application provided and how it should interact with the expert system of CARISMA, in order to perform a proper training and get a movement pattern for solar panels controlled under certain environmental circumstances. The expert system will be responsible for determining when it makes use of the neural network to determine the next move to make in a given solar panel. For this, our expert system will supply data environment and current position of panels to the network.

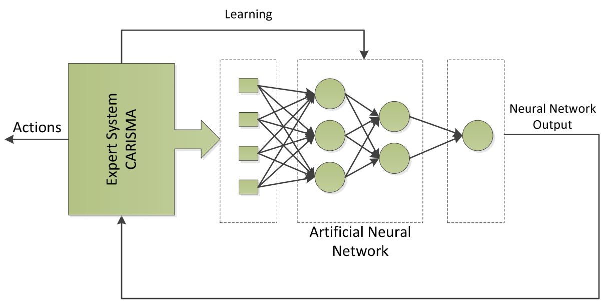


Fig. 4. Neural Network integration in CARISMA project

Additionally, the expert system will be responsible for neural network learning, comparing energy efficiency obtained from a particular movement. When output power energy from system is less due to a movement executed, the expert system will report the difference obtained to the neural network to adjust weights. If output power is incremented, then the expert system will not request new predictions from neural network, and it will set suitable weights for future requests.

V. Conclusions

The sun tracking systems have enabled uses of solar photovoltaic systems for complex applications. Compared to their traditional fixed-position, solar systems which track the changes in the trajectory of sun, generate a significantly higher output power. This paper has presented an integration model of an inference engine based on neural networks, along with a multiagent system, which allows the implementation of an adaptive control for tracking the sun.

Additionally, using our model of neural network in the design of a multiagent-based control system, we can obtain many benefits, such as adaptive control and simplicity. Problems in these control systems are scalability and flexibility: our model does not limit the number and types of agents and inclusion of new panels is possible, without additional cost.

Future lines of work include the final implementation of the model and testing to demonstrate its feasibility.

Acknowledgment

The work described in this paper has been funded by the Consejería de Innovación, Ciencia y Empresas (Junta de Andalucía) with reference number P08-TIC-03862 (CARISMA Project).

References

[1] Ali Al-Mohamad, “Efficiency improvements of photo-voltaic panels using a Sun-tracking system” in Applied Energy, Volume 79, Issue 3, 2004, pp. 345-354.

[2] Kajihara, A. and Harakawa, T., "Model of photovoltaic cell circuits under partial shading" in IEEE International Conference on Industrial Technology, 2005, pp.866-870

[3] Yang Jinhuan, Zou Qianlin, Tan Beiyue, et. al, “Photovoltaic Route Maps and Photovoltaic Power Generation Progresses of Various Countries”, in Solar Energy, vol.8, China, 2006, pp. 51.

[4] D. Oviedo, M. C. Romero-Ternero, M. Hernández, A. Carrasco, F. Sivianes, and J. Escudero, “Model of Knowledge Spreading for Multi-Agent Systems” in International Conference on Enterprise Information Systems. Iccs Conference Proceedings, 2010, pp. 326-331.

[5] S. C. Kremer, Field Guide to Dynamical Recurrent Networks, 1st ed., J. F. Kolen, Ed. Wiley-IEEE Press, 2001.

[6] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks”, Neural Computation, 1989.

[7] H.P. Garg and S.N. Garg, “Prediction of global solar radiation from bright sunshine hours and other meteorological data” in Energy Conversion and Management, Volume 23, Issue 2, 1983, pp. 113-118.

[8] SunTracking Prototype-Kit System, Suyero ITS Solutions. 2013 http://www.suntracking.es

[9] Hector M. Romero Ugalde, Jean-Claude Carmona, Victor M. Alvarado, Juan Reyes-Reyes, “Neural network design and model reduction approach for black box nonlinear system identification with reduced number of parameters”, Neurocomputing, Volume 101, 2013, pp. 170-180.

[10] R. F. Flórez and J. M. F. Fernández, “Las Redes Neuronales Artificiales: fundamentos teóricos y aplicaciones prácticas”, Netbiblo, 2008.

[11] Falas, T.; Stafylopatis, A.-G., "The impact of the error function selection in neural network-based classifiers", International Joint Conference on Neural Networks, 1999, vol.3, pp.1799-1804.

[12] D. Oviedo, M. C. Romero-Ternero, M. Hernández, A. Carrasco, F. Sivianes, and J. Escudero, “Architecture for Multiagent-Based Control Systems”, Berlín, Springer, 2010, vol. 79, pp. 97-104.