A neural network to predict reactor core behaviors postprint

Authors: Juan Jos Ortiz-Servin, David A. Pelta, Jos Alejandro Castillo

Date: 2023-06-18T00:00:00+00:00

Abstract

The global fuel management problem in BWRs (Boiling Water Reactors) can be understood as a very complex optimization problem, where the variables represent design decisions and the quality assessment of each solution is done through a complex and computational expensive simulation. This last aspect is the major impediment to perform an extensive exploration of the design space, mainly due to the time lost evaluating non promising solutions. In this work, we show how we can train a Multi-Layer Perceptron (MLP) to predict the reactor behavior for a given configuration. The trained MLP is able to evaluate the configurations immediately, thus allowing performing an exhaustive evaluation of the possible configurations derived from a stock of fuel lattices, fuel reload patterns and control rods patterns. For our particular problem, the number of configurations is approximately 7.71010; the evaluation with the core simulator would need above 200 years, while only 100 hours were required with our approach to discern between bad and good configurations. The later were then evaluated by the simulator and we confirm the MLP usefulness. The good core configurations reached the energy requirements, satisfied the safety parameter constrains and they could reduce uranium enrichment costs.

Full Text

Preamble

A Neural Network to Predict Reactor Core Behaviors

Juan José Ortiz-Servín,[1],† David A. Pelta,[2] and José Alejandro Castillo[1]
[1]Instituto Nacional de Investigaciones Nucleares, Carretera México-Toluca S/N, La Marquesa Ocoyoacac, Estado de México, CP 52750, Mexico
[2]ETS Ingeniería Informática y Telecomunicaciones, Universidad de Granada, C/Daniel Saucedo Aranda, s/n 18071, Granada, Spain

(Received October 15, 2013; accepted in revised form December 6, 2013; published online February 20, 2014)

The global fuel management problem in BWRs (Boiling Water Reactors) can be understood as a very complex optimization problem, where the variables represent design decisions and the quality assessment of each solution is done through a complex and computationally expensive simulation. This last aspect is the major impediment to performing an extensive exploration of the design space, mainly due to the time lost evaluating non-promising solutions.

In this work, we show how we can train a Multi-Layer Perceptron (MLP) to predict the reactor behavior for a given configuration. The trained MLP is able to evaluate the configurations immediately, thus allowing an exhaustive evaluation of the possible configurations derived from a stock of fuel lattices, fuel reload patterns, and control rod patterns. For our particular problem, the number of configurations is approximately 7.7×10^{10}; the evaluation with the core simulator would need above 200 years, while only 100 hours were required with our approach to discern between bad and good configurations. The latter were then evaluated by the simulator and we confirm the MLP usefulness. The good core configurations reached the energy requirements, satisfied the safety parameter constraints, and they could reduce uranium enrichment costs.

Keywords: Boiling Water Reactors (BWRs), Neural Networks, Optimization
DOI: 10.13538/j.1001-8042/nst.25.010602

Introduction

In a previous work [**?**], we presented a Recurrent Neural Network (RNN) to find good configurations from several stocks of optimized solutions to fuel lattice design, fuel load pattern design, and control rod patterns design. These partial solutions to the global fuel management problem are combined to find a core configuration of fresh fuel bundles, a fuel reload pattern, and core exposition calculus are made through control rod patterns in several burnup steps in the cycle length.

SIMULATE-3 [**?**] core simulator was used to calculate the reactor behavior of those configurations. Thermal limits, throughput of the cycle, and cold Shutdown Margin (SDM) at the beginning of the cycle are calculated in order to determine the quality of the configurations. Fuel lattices with several average uranium enrichments were used in the fuel lattice stock.

In the first instance (Ref. [**?**]), it was possible to find good configurations with average uranium enrichments lower than a reference case. Fuel lattices stock was created by Neural Networks (NN) [**?**] and Path Relinking [**?**] techniques. Both optimization techniques use CASMO4 [**?**] to calculate the lattice parameters: local power peaking factor and a reactivity value, both at the beginning of the fuel lattice life. Fuel reloads were generated using NN [**?**] and Tabu Search [**?**]. Both optimization techniques use SIMULATE-3 to calculate the end of cycle

under Haling condition [**?**]. Finally, control rod patterns were generated by Tabu Search [**?**] and Ant Colony System [**?**] optimization techniques. SIMULATE-3 was used to determine thermal limits throughout the cycle.

In this contribution our aim is to address the following research questions: (1) Is it possible to design a surrogate model of the simulator that allows performing a fast discrimination between good and bad configurations? (2) Having the previous model, would it be possible to evaluate the set of all the potential configurations arising from the combinations of alternatives in the stocks available?

In order to address these questions, we propose a Multi-Layer Perceptron (MLP) as a simplified model of the simulator to discern between bad or good core configurations. In Ref. [**?**], an adaptive classifier model is used to solve optimization problems. The classifier eliminates non-feasible solutions, reducing the CPU time to solve the problem. In our paper, the MLP eliminates bad core configurations.

The rest of the paper is organized as follows. In Sec. II we briefly describe the MLP concepts. Then in Sec. III, we show how the MLP was trained. In Sec. IV and Sec. V, we show some practical results with the trained MLP. Finally, conclusions and references are shown.

II. Multi-Layer Perceptron Neural Network

The artificial neural network is a computer model that can be used for pattern recognition, prediction, memory, etc. There are several neural network models, and one of the most popular is the Multi-Layer Perceptron (MLP) [**?**]. In Fig. 1 [Figure 1: see original paper], we show the typical architecture of this kind of neural network.

The neural network is composed of one input layer, one or more hidden layers, and one output layer. The input layer collects external information and distributes it to hidden layers. Hidden layers process the information, and the output layer shows results. Each layer has several neurons, where the neuron is the lowest information processor. The neuron makes a weighted sum of all its input signals:

$$I_i = \sum w_{ij} x_j$$

where I_i is the net input to the i-th neuron, w_{ij} is the weight connection between the i-th neuron (in a previous layer) and the j-th neuron (in a current layer), and x_j is the signal between both neurons. Then the net input is converted to an activation signal according to the activation function $f(I_i)$:

$$A_i = f(I_i)$$

The activation function gives a trigger threshold for the neuron. If the net input is lower than the threshold, the neuron is inhibited. If the net input is greater than the threshold, the neuron is excited. These inhibitory or excitatory signals are propagated by all neurons in the network until a global response is generated.

Weights connection between neurons must be adjusted in order for the MLP response to become adequate to the input signal. This process is named neural network training. Back propagation is the most popular training algorithm used for MLP. First, the input signal is passed through the layers until a response is generated. Second, the response is compared with the desired output and an error signal is generated. Third, the error signal is back-propagated to the first hidden layer, updating weight connections. The process is repeated until the error signal is lower than the tolerance.

III. Data Sets and Training Process

Neural network training was made using a set of 2680 samples. An explanation about how these samples were obtained will be shown in the next section. Each sample is a pair of input and output vectors where the former is a possible core reactor configuration (partial solutions to the global problem), and the latter is a set of core safety parameters (thermal limits, k_{eff} and SDM) which are calculated by SIMULATE-3 for that core reactor configuration. The values in the output vectors are aggregated into a single real value.

Input vectors: For this study, an 18-month equilibrium fuel cycle is used. The fuel reload has two fresh fuel batches. Both fresh fuel batches have a similar axial design: one node of natural uranium at the bottom, 8 nodes with 4.01% U-235 and variable gadolinia concentration, 6 nodes with 4.01% U-235 and high gadolinia concentration, 8 nodes with 3.96% U-235 and high gadolinia concentration, and finally two nodes of natural uranium at the top of the fuel bundle.

A variable number of fuel lattices for three segments of both fresh fuel batches were generated by Path Relinking and Neural Networks. Fuel reloads were generated using Neural Networks and Tabu Search. Control rod patterns were generated by Tabu Search and Ant Colony System optimization techniques.

Input vectors are represented by an 8-entry array. The first six entries are used to represent 3 axial segments of both fresh fuel batches. Entry number seven is used to specify a fuel loading pattern. Finally, the last entry is used to specify a set of control rod patterns throughout the cycle. For all entries, integer numbers are used to specify a fuel lattice, a fuel reload, or control rod patterns according to the list. An example of an input vector is shown in Fig. 2 [Figure 2: see original paper].

Each input vector is unique and defines a particular core reactor behavior. According to the size of lists used in this work, the universe of possible solutions

to this problem is:

$$33 \times 33 \times 17 \times 23 \times 17 \times 10 \times 56 \times 19 \approx 7.7 \times 10^{10}$$

Output vector: In order to construct the output vector, we proceed as follows. An input vector is introduced into SIMULATE-3 in order to do several runs and obtain thermal limits (MFLCPR, MFLPD, and MAPRAT), k_{eff} throughout the cycle, and cold shutdown margin at the beginning of the cycle. These core parameters are satisfied if they fulfill the following constraints:

1. Limiting fractions to Linear Heat Generation Rate (MFLPD) < 0.93
2. Limiting fractions to Critical Power Ratio (MFLCPR) < 0.93
3. Limiting fraction to Average Planar Linear Heat Generation Rate (MAPRAT) < 0.93
4. k_{eff} - target $k_{eff} < 400$ pcm (1 pcm $= 10^{-5}$)
5. Cold shutdown margin > 0.01

Some reactor core configurations may fulfill all or some of the safety parameters. For the purposes of this work, if a parameter is not fulfilled in only one burnup step, then it is considered not globally fulfilled. The same applies for thermal limits. Then, the number of safety parameters fulfilled can be determined for each core reactor configuration (input vector). The value in the output vector is calculated as a function of the number of core parameters fulfilled (CPF):

$$\text{Output Value} = e^{-(1-\text{CPF}/5)}$$

When CPF is equal to zero, Output Value is e^{-1}. When CPF is 5, Output Value is $e^{0} = 1$.

The MLP will predict the Output Value, and then we can use such prediction to calculate the number of core parameters fulfilled. The neural network has three layers: an input layer with 8 neurons, a hidden layer with 4 neurons, and the output layer with only one neuron. The number of neurons in hidden layers was determined by analyzing the neural network behavior for sizes: 3, 4, 5, and 6 neurons. The best results were obtained for 4 neurons, so that is the value kept for the rest of the paper.

IV. Experiments and Results

The 2680 samples in the dataset were divided into two subsets: training set and test set. The training set has 70% of all samples and the test set has the remaining 30%. Both subsets were randomly created from the original one. The MLP was trained with the back-propagation algorithm using the Generalized Delta Rule for weights updating. The program BackProp [**?**] was used to train the MLP.

Figs. 3 and 4 show the CPF values for training and test sets, respectively. As the MLP predicts the Output Value, the corresponding CPF is calculated using the inverse function of Eq. (3). Please note that CPF values calculated from MLP predictions are continuous values, while CPF values calculated from SIMULATE-3 are discrete values. In case of perfect learning, a 45° line should be observed in both figures. The results show that the MLP is able to roughly distinguish between very bad or very good configurations (those having low or high CPF values). In other words, the MLP almost never classified a very good configuration (according to SIMULATE-3) as a very bad configuration.

To further analyze the results, we will consider the following questions: How many core configurations are recognized as good configurations by MLP but SIMULATE-3 says they are bad? And, in turn, how many good configurations (according to SIMULATE-3) would be discarded by the MLP? If we take a threshold value like 4.5 in the MLP' s predicted Output Value, we can consider as "good" configurations those that are above the threshold and as "bad" those that are below. Truly good configurations are those with CPF = 5 according to SIMULATE-3.

Now, taking the problem as a binary classification one, Table 1 shows the so-called Confusion Matrix for training and test sets. This matrix indicates the amount of True Positives (TP, good core configurations according to both SIMULATE-3 and MLP), True Negatives (TN, bad core configurations according to both SIMULATE-3 and MLP), False Negatives (FN, good core configurations according to SIMULATE-3 but MLP classifies them as bad), and False Positives (FP, bad core configurations according to SIMULATE-3 but MLP classifies them as good) obtained.

The MLP learned to classify core configurations with acceptable confidence. As shown in Table 1 , the number of False Negatives is high, meaning that MLP could discard an important number of good core configurations. On the other hand, a low number of False Positives means that few bad core configurations are considered as good core configurations.

Two additional statistical measures of the performance of a binary classification test can be calculated from the confusion matrix: sensitivity and specificity. Their definitions are:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Sensitivity (also called recall rate in some fields) measures the proportion of actual positives that are correctly identified as such. Specificity measures the proportion of negatives that are correctly identified. A perfect predictor would

be described as having 100% sensitivity and 100% specificity. In our case, the results are as follows:

TABLE 2 . Sensitivity and Specificity of the binary classification based on the NN

Set	Sensitivity (%)	Specificity (%)
Training set	62.35	99.32
Test set	60.00	98.50

In other words, the MLP is excellent for distinguishing bad configurations but not so good at detecting the good ones. However, we should recall that our aim is to explore the whole universe of solutions and, in order to do this, we should avoid the full evaluation (with SIMULATE-3) of bad or not promising configurations. This process is described in the next section.

V. MLP Used to Find Good Core Configurations

The trained MLP was used as a filter in the process of exhaustive enumeration of possible configurations. Given a configuration, we evaluate it with the MLP, and if the predicted output value is greater than a certain threshold, then the configuration is considered potentially good and is archived for later evaluation with SIMULATE-3.

Table 3 shows, for a given threshold value, the number of core configurations that were considered potentially good, how many of them were effectively good (according to SIMULATE-3), and the relation between both values.

TABLE 3. Results of the exhaustive enumeration process. The total number of core configurations was 7.7×10^{10}

Threshold Value	Core configurations with Output Value > threshold (A)	Core configurations with CPF = 5 according to SIMULATE-3 (B)	(B/A) × 100 %
4.5	580,000	1,141	0.20
4.7	110,000	1,141	1.04
4.9	20,000	1,141	5.71

Using a threshold value of 4.5 gave around 580,000 core configurations, with less than 1% of them being effectively good. Using a higher threshold effectively reduced the configurations that passed the filter and increased the percentage of effectively good configurations.

VI. Core Configurations Analysis

In this section we will analyze the configurations obtained after the enumeration process. The study was made for an equilibrium BWR cycle of 18 months, with a cycle length of 10,896 MWD/T at full power conditions. For this cycle exposure, the target k_{eff} is set to 0.9978.

The fuel reload has two fresh fuel bundle batches. The first one (Batch A) has an average uranium enrichment of 3.66%, 10 gadolinia rods, and a batch size of 60 fuel bundles. The second batch (Batch B) has the same average uranium enrichment, 8 gadolinia rods, and a batch size of 52 fuel bundles. A uranium requirement (UR) for this fuel reload can be defined as:

$$\text{UR} = 60 \times \text{UFB-A} + 52 \times \text{UFB-B}$$

where UFB-A is the average uranium enrichment for fuel Batch A and UFB-B is the average uranium enrichment for fuel Batch B.

The baseline for comparison is a reference core configuration with UR = 409.92% (applying Eq. (6) and k_{eff}-EOC = 0.9978). Core configurations with lower UR values mean they save uranium with respect to the reference one.

From the results shown in Table 3, we have available 1,141 core configurations with CPF = 5. From this set, we will consider only 165 configurations having a k_{eff}-EOC greater than the reference value.

Figure 5 [Figure 5: see original paper] shows a scatter plot where each point represents a core configuration. The X-axis is the uranium saving (according to Eq. (6) and the reference UR = 409.92%), while the Y-axis indicates the difference against the k_{eff} reference value.

It is clear from the plot that there are better configurations than the reference one, both improving k_{eff} and uranium savings. Core configurations above the solid line and marked with solid squares are those that decrease the uranium enrichment of one of the fresh fuel batches without loss of energy production. These core configurations could have economical advantages with respect to the reference one, both through uranium savings and electrical energy sales. Core configurations under the line (in solid circles) are good core configurations without economical advantages with respect to the reference one.

The core configurations analysis can also be done from other points of view. First, we measured the Hamming distance [**?**] between the reference core configuration and the selected ones. We obtained 35 configurations with one position changed, 122 with two positions changed, and 8 configurations with three changes.

Then, in order to analyze where those changes happened (i.e., what are the necessary changes in the reference configuration to obtain a better solution), we counted for every entry the number of configurations with a different value than

that of the reference configuration (we must remember that the reference configuration is [1,1,1,1,1,1,1,1]). The maximum value for an entry is 165, stating that all configurations under analysis have a different value than the reference one. The results are shown in Table 4 .

TABLE 4 . Hamming distances between the reference configuration and the selected ones

Entry	Configs	% (over 165)
1	0	0.0
2	160	97.0
3	0	0.0
4	0	0.0
5	160	97.0
6	0	0.0
7	165	100.0
8	0	0.0

It is clear that the entries with greater variability are 2, 5, and 7; there were no configurations with alternatives other than the reference one for entries 6 and 8. Table 4 shows that the reference fuel lattice number two of both fuel batches can be improved. Almost all good core configurations have different fuel lattices in both batches. Also, both fuel lattices are responsible for the uranium savings. More energy production is due to the use of another fuel reload different from the reference one.

VII. Conclusion

From this work, several comments can be made:

- It was possible to train an MLP able to catalog good core configurations. A core configuration is a combination of fuel lattices, control rod patterns throughout the cycle, and a fuel reload.
- The MLP training required around one hour to learn the desired behavior on an AMD processor at 2.1 GHz with 1.5 GB RAM. This small time interval makes the trained MLP an excellent tool to be coupled with an optimization system to find the best core configuration.
- The trained MLP could be used as a "bad configuration filter" to avoid running a time-expensive 3D core simulator.
- The utility and quality of the trained MLP was demonstrated by an analysis of results in both test and training sets and their performance when coupled with an exhaustive searching algorithm.
- A balance between the required time to evaluate core configurations and the analyzed solution space size can be made by adjusting the filter threshold. Lower threshold values enable the coupled system to explore more

solutions in the optimization process. High threshold values reduce the required CPU time to find a good core configuration.

- This coupled system found several good core configurations with advantages: some overcome the energy requirements with the same uranium enrichment as the reference case, while others reach the energy requirements while decreasing uranium enrichment. Several core configurations satisfy both scenarios.
- Table 4 tells us about the performance of optimization designs. We can say that we are able to design fuel reloads, treating the problem as an independent one, with acceptable confidence. Similar ideas can be said for fuel lattice design. On the other hand, control rod pattern designs have poor performance when they are optimized as an independent problem separate from the rest of the integral optimization.

References

[?] Ortiz J J, Castillo J A, Pelta D A. Nucl Eng Des. 2011, 241: 1519-1527.
[?] Dean D W. SIMULATE-3 Advanced Three-Dimensional Two-Group Reactor Analysis Code, SSP-95/15 - Rev 3, Studsvik Scandpower, 2005.
[?] Ortiz J J, Castillo A, Montes J L, et al. Nucl Sci Eng, 2009, 162: 169-177.
[?] Castillo A, Ortiz J J, Perusquía R, et al. Prog Nucl Energ, 2011, 53: 368-374.
[?] Rhodes J and Edenius M. CASMO-4 A Fuel Assembly Burnup Program, SSP-01/400 Rev 4. Studsvik Scandpower, 2004.
[?] Ortiz J J and Requena I. Ann Nucl Energy, 2004, 31: 789-803.
[?] Castillo J A, Alonso G, Morales L B, et al. Ann Nucl Energy, 2004, 31: 151-161.
[?] Haling R K. Operational Strategy for Maintaining an Optimum Power Distribution through Core Life. Proc. ANS Topl. Mtg. Nuclear Performance of Core Power Reactors, TID-7672. US Atomic Energy Commission, 1964.
[?] Castillo A, Ortiz J J, Alonso G, et al. Ann Nucl Energy, 2005, 32: 741-754.
[?] Ortiz J J and Requena I. Ann Nucl Energy, 2006, 33: 30-36.
[?] Tenne Y. Eng App Artif Intel. 2012, 25: 1009-1021.
[?] Caudill M and Buttler C. Understanding neural networks: Computer explorations. V. 1 Basic network. MIT Press (USA), 1994.
[?] Tveter D R. User Manual of Student Version Basis of AI Backprop. Copyright (c) 1990-97.
[?] Zwillinger D (Editor). Standard Mathematical Tables and Formulae. Chapman & Hall/CRC New York (USA), 2003.

Note: Figure translations are in progress. See original paper for figures.