PROCEEDINGS OF SCIENCE

Evolutionary Neural Network Modeling for Energy Prediction of Cloud Data Centers

Yong Wee FOO[1]
School of Engineering, Nanyang Polytechnic, Singapore
School of Engineering, University of Glasgow, Glasgow G12 8QQ
E-mail: Foo_Yong_Wee@nyp.edu.sg

Cindy GOH
School of Engineering, University of Glasgow, Glasgow G12 8QQ
E-mail: Cindy.Goh@glasgow.ac.uk

Hong Chee LIM
School of Engineering, University of Glasgow, Glasgow G12 8QQ
2110174L@student.gla.ac.uk

Yun LI
School of Engineering, University of Glasgow, Glasgow G12 8QQ
E-mail: Yun.Li@glasgow.ac.uk

Accurate forecasts of data center energy consumptions can help eliminate risks caused by under-provisioning or waste caused by over-provisioning. However, due to nonlinearity and complexity, energy prediction remains a challenge. An added layer of complexity further comes from dynamically changing workloads. There is a lack of physical principle based clear-box models, and existing black-box based methods such neural networks are restrictive. In this paper, we develop an evolutionary neural network as a structurally optimal black-box model to forecast the energy consumption of a dynamic cloud data center. In particular, the approach to evolving an optimal network is developed from several novel mechanisms of a genetic algorithm, such as a structurally-inclusive matrix encoding and species parallelism that help maintain an overall increasing fitness to overcome slow convergence whilst preventing premature dominance. The model is trained using part of the data obtained from a set of MapReduce jobs on a 120-core Hadoop cluster and is then validated against unseen data. The results, both in terms of prediction speed and accuracy, suggest that this evolutionary neural network approach to cloud data center forecast is highly promising.

International Symposium on Grids and Clouds (ISGC) 2015
15 -20 March 2015,
Academia Sinica, Taipei, Taiwan

[1] Speaker

1. Introduction

The proliferation of cloud computing has meant that energy demand for data centers continues to multiply. A more efficient use of energy can prevent this type of increased consumption from spiraling out of control. A more sustainable data center also makes business sense as it improves profitability and reduces environmental impact. To help achieve this, more accurate energy prediction will play a key role, as it helps data center owners adjust and schedule resources accordingly. With this motivation, we set out to develop an evolutionary neural network model with the aim to provide a fast and accurate energy forecast for cloud data centers [1] [2].

Traditionally, the training of neural networks (NNs) uses a method called backpropagation, which is based on gradient guidance to optimize the network weights with a fixed architecture to produce the lowest modeling error. Albeit the efficiency of this approach, backpropagation often results in the search being trapped in a local minimum. Moreover, due to a fixed number of neurons and connections, this approach is unsuitable for modeling non-linear systems, where multiple local minima often exist. Within a data center, multiple electrical and mechanical systems interact in a complex manner in supplying power and cooling to the physical servers consolidated within virtual machine cloud environment. Hence, the energy consumption characteristics of cloud data centers exhibit non-linearity [3]. A global search algorithm such as the genetic algorithm (GA) overcomes this difficulty, but suffers from low convergence as it is a nondeterministic polynomial algorithm.

In this paper, a novel implementation mechanism of the GA is proposed to improve the convergence rate for NN modeling. In particular, a new encoding scheme embodies the NN weights and structure in a matrix representation, so as to enhance the efficiency of the GA operations of crossover and mutation for the NN search. Further, sub-populations are used for individuals in each species to thrive through intra-species crossover, protecting the species from pre-mature extinction before they have a chance to evolve into healthier individuals. To train and validate such an evolutionary NN model, we set up a Hadoop cluster for our experiments.

The rest of the paper is organized as follows. In Section 2, NN modeling for data center energy consumption is formulated. The evolutionary approach is then detailed in Section 3. Experimental results are presented and examined in Section 4, based on data collected from the Hadoop cluster. In Section 5, conclusions are drawn and future work is highlighted.

2. Neural Network Modeling

An artificial neural network mimics the way the human brain learns and recognizes patterns. NN models are optimized against training data often collected from experiments [5] [6]. In our modeling, data are collected from a Hadoop cluster.

2.1 Network Structure

Figure 1 shows a typical three layer (or single hidden layer) feed-forward Multi-Layer Perceptron (MLP) NN [7]. In this work, the number of neurons is set to a maximum of 20 for the hidden layer and the number of hidden layer fixed to 1. Based on the universal approximation theorem [13], any arbitrary continuous function can be arbitrary well approximated by such MLP. In [11], NN with 1 hidden layer has been utilized to successfully forecast the photovoltaic power yield. In Figure 1, the structure of a NN consists of a constellation of interconnected neurons. These neurons send unidirectional signals to other neurons via these interconnections. The strength of a signal received by a neuron is dependent on the connection weight where the signal is being carried along. The more excitatory the

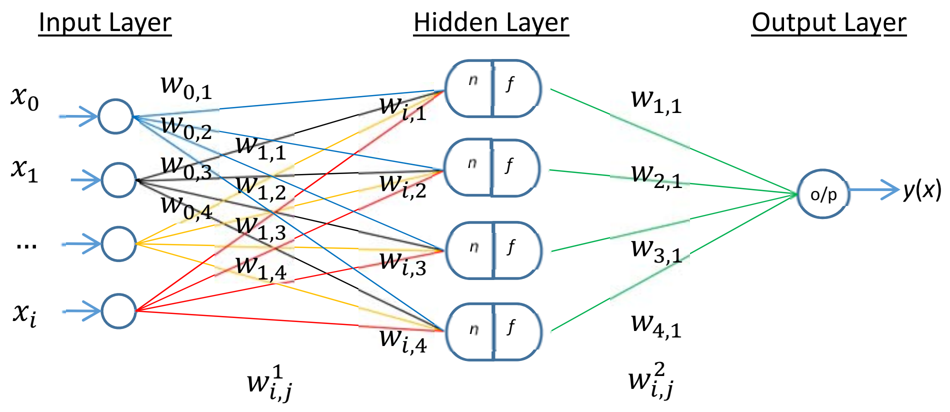


Figure 1: A multi-layer perceptron neural network with three layers

weight, the more amplified the signal. The sum of the weighted signals received by the neuron is then put through an activation function to normalize the amplitude of the output of the neuron. In Figure 1, x_i represents the inputs, $w_{i,j}$represents the weights matrix between input *i* and neuron *j*. Each *n* is the sum of the products of all the inputs with their respective weights as expressed in (1),

$$n = \sum_j w_{i,j}\ x_i \tag{1}$$

The activation function is represented by *f*. An example of this is the uni-polar sigmoid function given by (2).

$$f(n) = \frac{1}{1+e^{-n}} \tag{2}$$

The output *y*(*x*) is the result predicted by the entire NN at the output layer where,

$$y(x) = \sum_j w_{i,j}\ .f(n) \tag{3}$$

2.2 Training Neural Networks

NN training utilizes learning algorithms to improve the network parameters such as weights and structure using empirical inputs and output data of an observed system that is to be modeled. The inputs represent energy related features collected from the Hadoop cluster and the output represents the energy consumption of the cluster. The neural network features are listed as follows:

Neural Network Features:

1. Number of Map and Reduce Instructions
2. CPU Utilization (%)
3. System Load (%)
4. Memory Use (%)
5. Map Read (Gigabyte)
6. Reduce Read (Gigabyte)
7. Map Write (Gigabyte)
8. Reduce Write (Gigabyte)
9. Reduce Shuffle (Gigabyte)
10. Network Bandwidth (Gigabit per second)
11. File size (Gigabyte)
12. Job Completion Duration (Hour)

Neural Network output:

1. Energy consumption (kilowatt-hour)

The features, along with their method of collection, are described in Table 1 below.

Category		Metric	Unit	Description	Method of collection
Inputs	CPU/ System	1. Number of Map and Reduce Instructions	Number	Job's instruction number	Ganglia
		2. CPU utilization	%	Percentage of CPU time process the MapReduce process	Ganglia
		3. System Load	Number	System capacity in processing MapReduce workload	Ganglia
	Memory	4. Memory use	%	Percentage of memory use during the MapReduce process	Ganglia
	Disk IO	5. Map read 6. Reduce read 7. Map write 8. Reduce write	Gigabyte Gigabyte Gigabyte Gigabyte	Data read by Map from local disk Data read by Reduce from local disk Data written by Map to local disk Data written by Reduce to local disk	Hadoop built-in counters
	Network	9. Reduce Shuffle bytes	Gigabyte	Data transferred from Map to Reduce	Hadoop built-in counters
		10. Network Bandwidth	Gigabit per sec	Data transmitted and received	Ganglia
	Job Profile	11. File size	Gigabyte	Size of MapReduce jobs	Hadoop built-in counters
		12. Job completion duration	Hour	Time taken to finish a MapReduce job	Hadoop built-in counters
Output	Energy	1. Energy consumption	kWh	Energy consumed by Hadoop cluster	SNMP retrieve from Raritan iPDU

Table 1 Energy-Related Metrics for Hadoop Cluster

The learning of a NN generally follows the gradient of an error function through backpropagation. As gradient guidance often converges to a local minima [7] and cannot be used to determine an optimal network structure, we propose the use of a GA to evolve both the structure and the weights of the NN for the energy consumption modeling of cloud data center.

3. Evolutionary Neural Network Modeling

The evolutionary NN modeling technique being developed in this paper aims to provide an accurate and computationally efficient approach to the optimization of both the weights and the structure of the NN during the training phase. There are up to 12 neurons for the input layer, with an additional input of a unity value to represent a bias term through the weight to be trained for this input. The output layer comprises of the single neuron, representing the energy consumption. The number of layers is fixed at 3. Although this can be varied to model various complex systems, it is proven that a single hidden layer is sufficient to approximate any arbitrary function and the general consensus is that any improvements in the efficiency of hidden neurons are typically very small [8]. As an empirically-derived rule-of-thumb suggests that an optimal number of hidden neurons is usually between that of the input and output ones [8], we set a maximum number of hidden neurons to 20 for the GA to optimize. The following sections describe in detail the GA encoding scheme and the evolutionary NN.

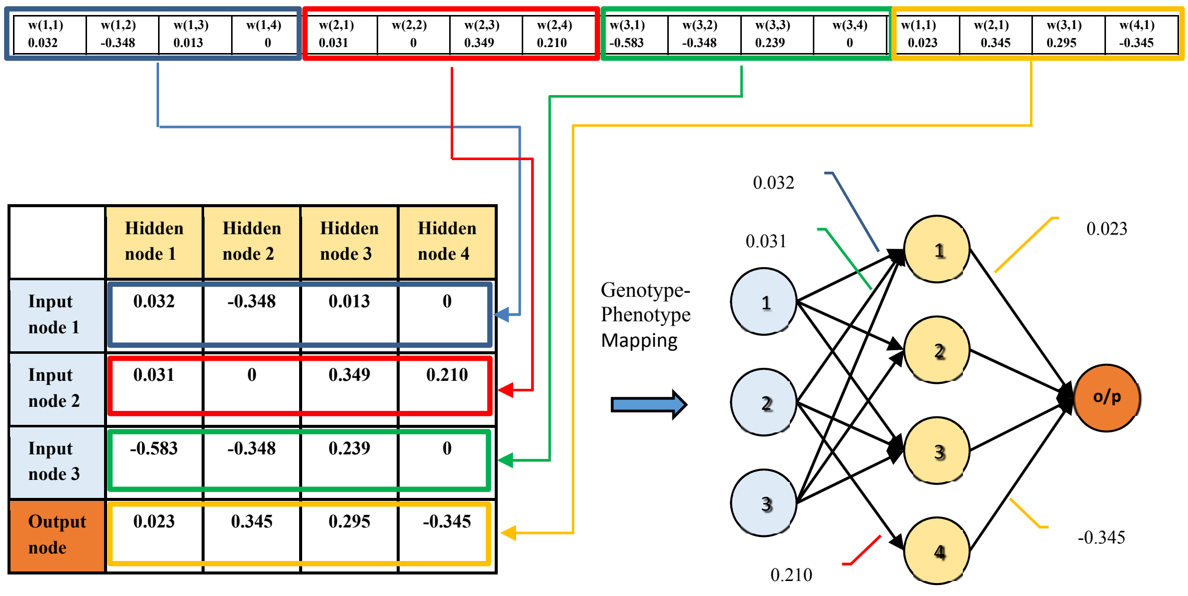
3.1 Neural Network Encoding

To encode the NN structure and weights, a direct encoding scheme is developed. This approach is able to represent a large class of NN solutions as well as to provide a stable search space for improved convergence. Figure 2 shows the chromosome for a NN with 3 input nodes, 4 hidden nodes and 1 output node. The length of the chromosome is determined by the maximum number of connections between the input, hidden and output nodes. In this case, it would be 16 where the maximum number of connections between the input and hidden nodes is 12 (3 inputs x 4 hidden neurons) and the maximum number of connections between the hidden and the output node is 4 (4 hidden nodes x 1 output node). The first 12 genes represents the connections and weights between input node *i* to hidden node *j,* denoted by *w(i,j)*. The last 4 genes represent the connections and weights between the hidden node *j* to output node 1, denoted by *w(j,1)*. A value of ‘0’ in the gene indicates the absence of a connection between the nodes. This chromosome can be easily expressed as a 4 x 4 matrix as shown in Figure 3a.

w(1,1)	**w(1,2)**	**w(1,3)**	**w(1,4)**	**w(2,1)**	**w(2,2)**	**w(2,3)**	**w(2,4)**	**w(3,1)**	**w(3,2)**	**w(3,3)**	**w(3,4)**	**w(1,1)**	**w(2,1)**	**w(3,1)**	**w(4,1)**
0.032	**-0.348**	**0.013**	**0**	**0.031**	**0**	**0.349**	**0.210**	**-0.583**	**-0.348**	**0.239**	**0**	**0.023**	**0.345**	**0.295**	**-0.345**

Figure 2. The Chromosome of the 3 Layer NN Model with 3 Input Nodes, 4 Hidden Nodes and 1 Output Node

Figure 3 shows the complete genotype-phenotype mapping with rows denoting input and output node connections and columns denoting hidden node connections. Input node 1 has connections to hidden node 1, 2 and 3 with weights 0.032, -0.348 and 0.013, respectively. All 4 hidden nodes are connected to the output node with weights 0.023, 0.345, 0.295 and -0.345, respectively. A weight value of ‘0’ indicates there is no connection between input node 2 and hidden 2 nodes.



w(1,1)	w(1,2)	w(1,3)	w(1,4)	w(2,1)	w(2,2)	w(2,3)	w(2,4)	w(3,1)	w(3,2)	w(3,3)	w(3,4)	w(1,1)	w(2,1)	w(3,1)	w(4,1)
0.032	-0.348	0.013	0	0.031	0	0.349	0.210	-0.583	-0.348	0.239	0	0.023	0.345	0.295	-0.345

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	0.032	-0.348	0.013	0
Input node 2	0.031	0	0.349	0.210
Input node 3	-0.583	-0.348	0.239	0
Output node	0.023	0.345	0.295	-0.345

(a). Genotype (b). Corresponding NN

Figure 3. Illustration of the NN encoding scheme and genotype-phenotype mapping.

3.2 GA Framework

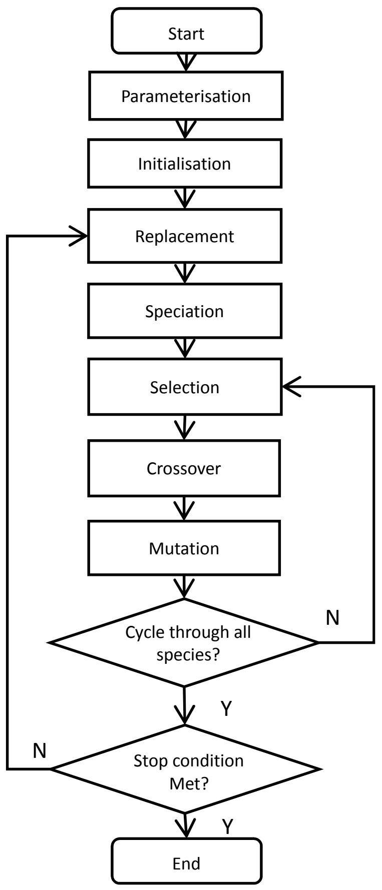
Figure 4 shows the framework and the corresponding pseudo code for the GA used in the NN modeling. The control parameters used in the GA are summarized in Table 2. The algorithm is briefly explained below.

The evolutionary process begins with the random initialization of individuals. This is the initial (global) population. At the start of each generational cycle, the fitness of the individuals is calculated and the weakest ones are replaced with the fittest. The “kill-percentage” determines the amount of individuals that will be replaced. The replacement strategy keeps the average population fitness on a progressive course by purging of the weaklings.

The next process is speciation. The speciation algorithm splits the population into sub-populations (see Figure 5). The compatibility consideration for speciation is based on the number of hidden nodes for a NN. For example, NN with 10 hidden nodes are grouped into a species by themselves called NN_10. NN with 12 hidden nodes are grouped into another species, NN_12, so on and so forth. Individuals compete for survival primarily within their own species instead of with the population at large. The objective of speciation is to grant time for individuals to evolve to healthier organisms without the threat of any one species taking over [14]. This way, premature extinction is withheld thereby preserving structural innovation and giving potential solutions within the species a chance to thrive. From the optimization viewpoint, each species contain a local optima [15] from which a global solution can be identified.

GA Control Parameters	**Description**	**Values**
pop_size	The size of the population.	100
max_gen	The maximum number of generational cycles.	100
xo_rate_intra	The intra-species crossover rate. This rate governs the percentage of new individuals in the respective species that will replace the old.	0.9
xo_rate_inter	The inter-species crossover rate. This rate governs the probability where one of the parent for crossover will come from another species.	0.01
mu_rate_wt	The mutation rate for the connection weights. This rate governs the probability where connection weights will be mutated.	0.015
mu_range_wt	The mutation range for the connection weights. This range governs the mutated weight value that will fall within this range centered on its original value.	-0.5 to 0.5
mu_wt_cap	The weight cap number that caps the mutated weight to 1.	1
mu_rate_conn	The mutation rate for re-enabling a connection. This rate governs the probability that a disabled link will be re-enabled.	0.01
kill_percentage	The kill percentage number. This number governs the percentage of the weakest individuals in the population will be replaced with the healthiest ones.	0.05

Table 2 GA Control Parameters and Values



```
Genetic Algorithm ()
[
Parameterise control data;
Initialise population;
do
  [
  Replace;
  identify and group species;
  do
    [
    select solution for next generation
    perform crossover
    perform mutation
    ]
  while cycle through all species not completed
]
while stop condition not met
]
end
```

Figure 4. Framework and corresponding pseudo code of the GA for NN modeling.

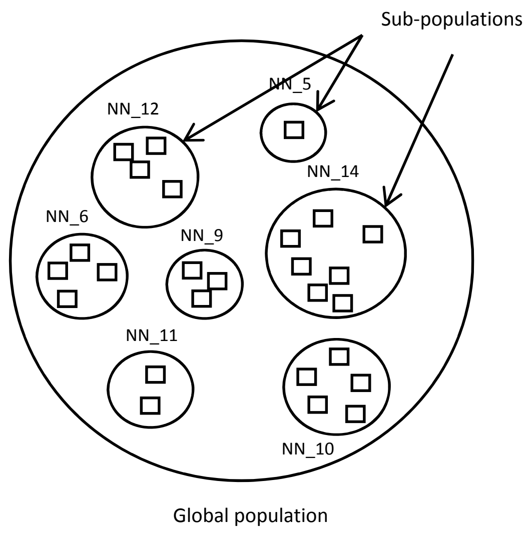


Figure 5. Speciation Forming Sub-population

After the speciation process, the selection process begins in the preparation for crossover. Here, the Stochastic Universal Sampling (SUS) selection scheme is applied to choose suitable candidates for crossover. Figure 6 provides an illustration of the SUS. In comparison to the Roulette-Wheel Selection (RWS) which provides no bias but does not guarantee a minimum spread, SUS is non-bias and ensures a minimum spread is maintained [9]. This is achieved by using a single random value to select the candidates at equally spaced intervals as opposed to the RWS where the apportioning is based on an individual's fitness such that the fittest individual has a highest chance of being selected In SUS, weaker individuals of the sub-population has equal chances to be chosen and therefore not allow the fittest individuals to dominate the candidate space prematurely.

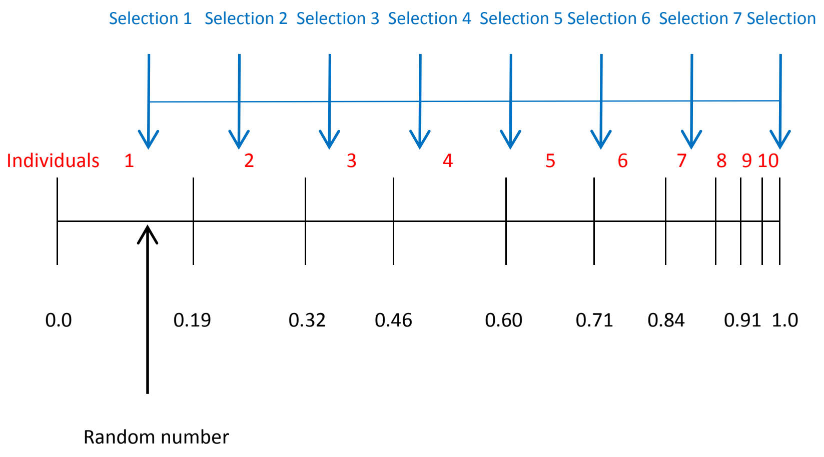


Figure 6. Stochastic Universal Sampling

Without loss of generality, the crossover rate is fixed at 0.9, for 90% of the population to be replaced with new offspring. We implement two types of crossover in this algorithm - intra-species crossover and inter-species crossover. In intra-species crossover, we apply a single-point crossover at the mid-point of the hidden layer in the NN structure. The parents selected for the crossover is provided by the SUS selection scheme. The portions of the parents' genome at the crossover point are recombined to produce the new offspring (see Figure 7). This method of crossover will always produce an offspring that retains the species' compatibility which allows that new individual to stay and thrive within the species. The safeguard is necessary to protect structural innovation and prevent the threat of any one species from taking over the entire population. The intra-species crossover rate is set to 0.9. Hence, 9 out of 10 individuals in their own species will be replaced with new offspring in each generational cycle. For inter-species, the crossover rate is set at 0.01. Here, we too apply a single-point crossover method however, this time it is at the mid-point of the genome, as shown in Figure 8. The recombination at the mid-point of the genome ensures new species can be formed thereby enhancing the search space. The parents for inter-species crossover are selected at random from the entire population.

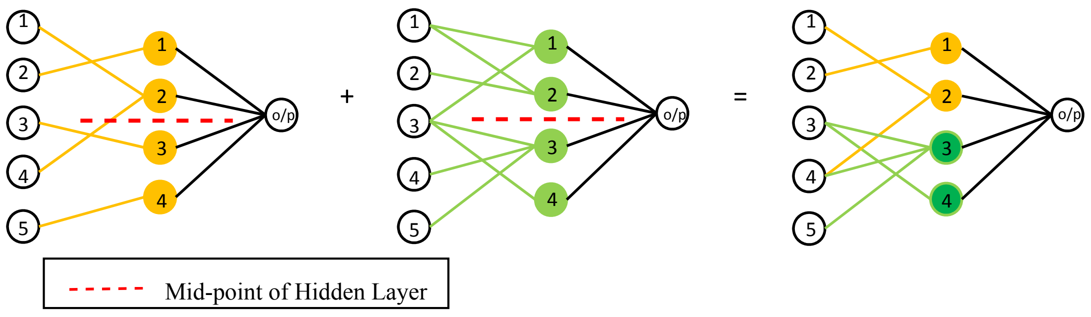
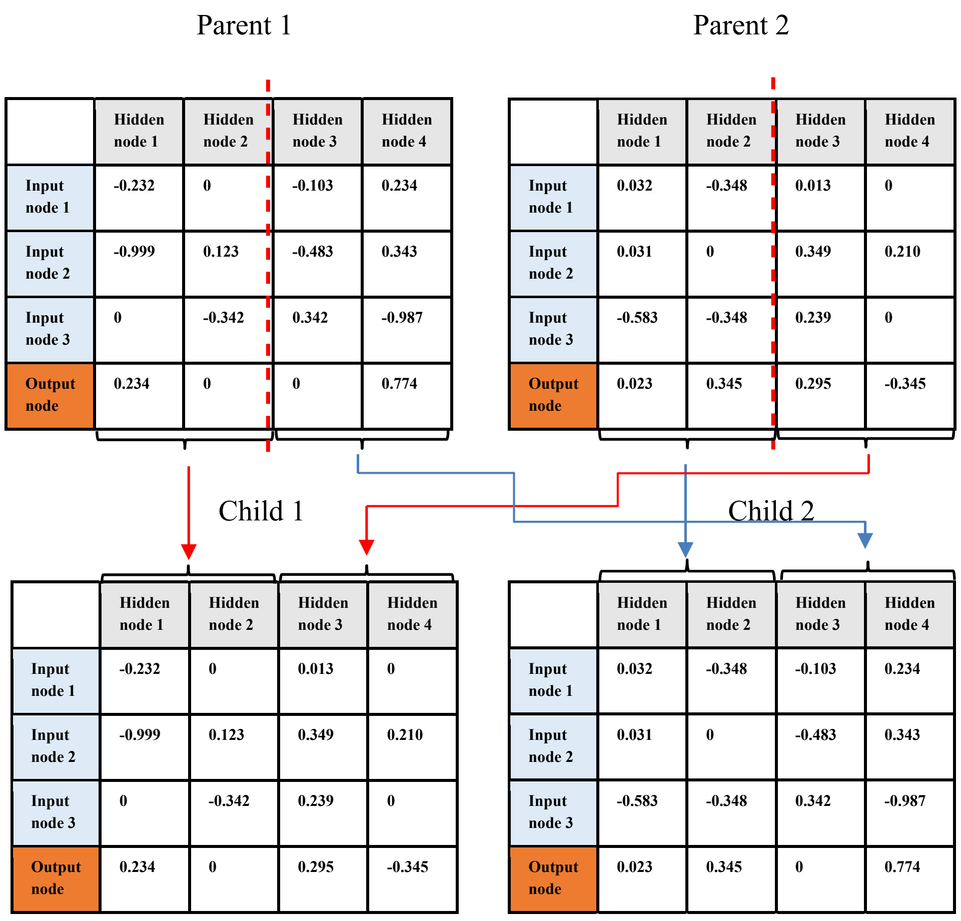


Figure 7. Intra-Species Crossover

The mutation rate for weights and connections are set to 0.015 and 0.01, respectively, where every individual has a 1.5% chance for its weights and 1% for structure to be changed during the mutation process. Structure mutation rate is lower of the two as this mutation, if occur, can effectively change the individual from one species to another. At the completion of the sub-population renewal on a global basis, the stop condition is assessed. If the stop conditions are not met – either the individual fitness > 0.99 or generational cycle reaches 100, the generational cycle repeats.



Parent 1

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	-0.232	0	-0.103	0.234
Input node 2	-0.999	0.123	-0.483	0.343
Input node 3	0	-0.342	0.342	-0.987
Output node	0.234	0	0	0.774

Parent 2

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	0.032	-0.348	0.013	0
Input node 2	0.031	0	0.349	0.210
Input node 3	-0.583	-0.348	0.239	0
Output node	0.023	0.345	0.295	-0.345

Child 1

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	-0.232	0	0.013	0
Input node 2	-0.999	0.123	0.349	0.210
Input node 3	0	-0.342	0.239	0
Output node	0.234	0	0.295	-0.345

Child 2

	Hidden node 1	Hidden node 2	Hidden node 3	Hidden node 4
Input node 1	0.032	-0.348	-0.103	0.234
Input node 2	0.031	0	-0.483	0.343
Input node 3	-0.583	-0.348	0.342	-0.987
Output node	0.023	0.345	0	0.774

Figure 8 Inter-Species Crossover

The generational cycle continues until the termination conditions are met. The search will terminate when either an individual's fitness within the population has reached the given target value of 0.99 or a fixed number of generational cycle, that is, 100 generations has passed.

The dataset is apportioned 70% for training and 30% for validation. The cost function J_{MSE} is a Mean Square Error (MSE), i.e. the average squared difference between the actual output, *a* and target, *t* as shown in Equation (4).

$$J_{\mathrm{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(t_i - a_i)^2 \qquad (4)$$

4. Results and Discussions

Figure 9 shows the fitness plot of the evolutionary NN. The fittest individual has a fitness of 0.9910 found in the 63th generation. The output plot can be seen in Figure 10 as the actual data (blue) track and try to fit the desired data (red). Figure 11 shows the solution structure of the fittest individual. It optimizes hidden layer to 9 neurons.

We validate the solution with dataset not used by the training. Figure 12 shows the validation plot by the NN solution. The blue graph has successfully predicted the red, which is the desired output with an MSE of 0.096216.

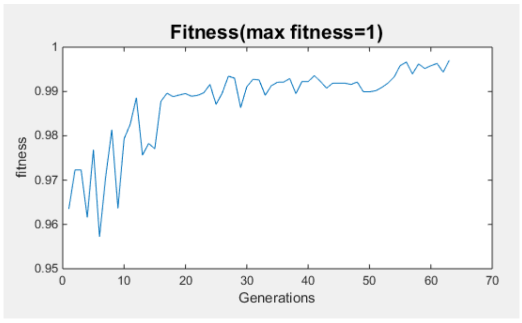


Figure 9. Fittest Plot

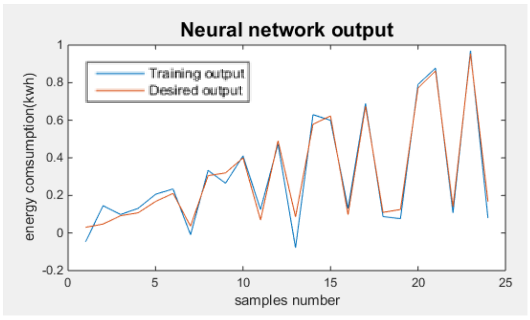


Figure 10. Training Plot (70% of Dataset)

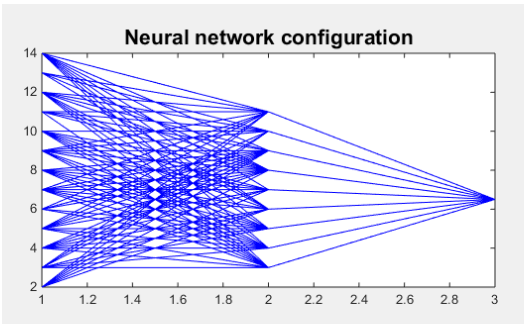


Figure 11. Optimized NN Structure

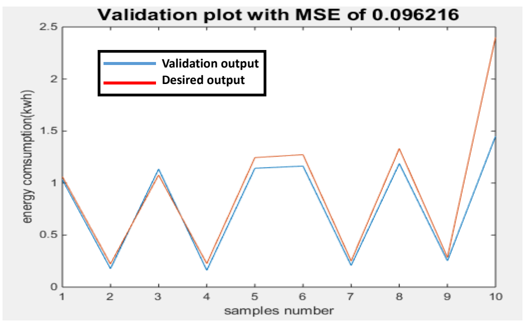


Figure 12. Validation Plot (30% of Unseen Dataset)

Figures 13 to 15 show the species or sub-population distribution at various stages of the generational cycle. At initialization (Figure 13), the population shows a normal distribution. Sub-populations NN_9 and NN_10 are most populous (23 count each) followed by NN_8 (13 count). In NN_4 and NN_16, there are only one individual respectively. NN_1, NN_2, NN_3, NN_17, NN_18, NN_19 and NN_20 do not exist. After 10 generations (Figure 14), the sub-population distribution changes. NN_9 size remains unchanged, NN_10 decreases slightly (20 count), NN_13 grow from a size of 6 to 11 while NN_4, NN_15 and NN_16 have extinct. After 20 generations (Figure 15), NN_9 has increased to 28, NN_6 has extinct. At the 63rd generation (Figure 16), the solution converged with the fittest individual located in NN_9. It is interesting to note that NN_6 and NN_16, previously extinct, these species have re-emerged and a new species NN_18 is formed. This shows that the algorithm search space has a good diversity.

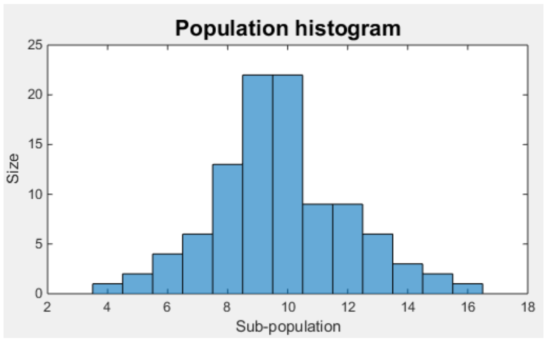


Figure 13. Initial Sub-population Distribution

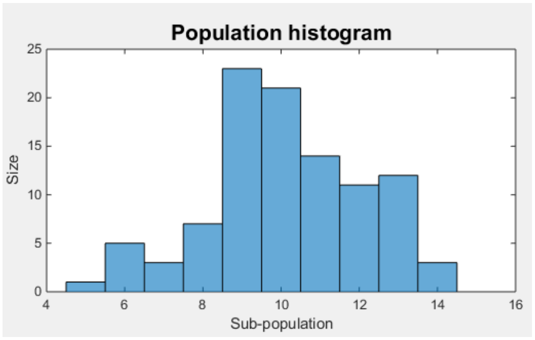


Figure 14. Sub-population Distribution after 10 Generations

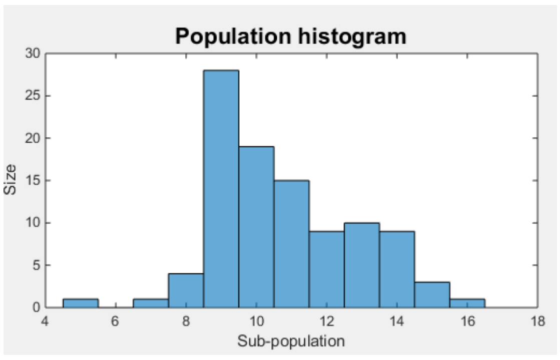


Figure 15. Sub-population Distribution after 20 Generations

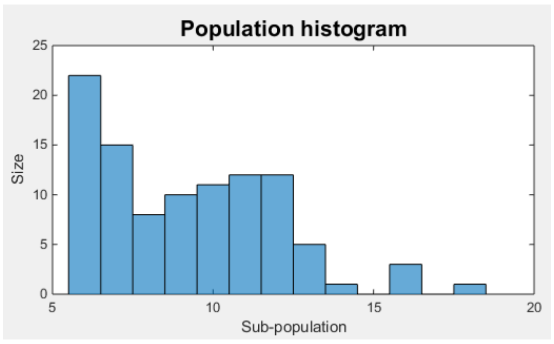


Figure 16. Sub-population Distribution at 63rd Generation (Converged) with Stop Condition of 0.997

To further investigate the implementation of the GA, in particular the impact of speciation and replacement strategy on the convergence rate and MSE, we performed an experiment to run the algorithm 20 times. Each time the combinations of speciation and replacement strategy is varied as shown in Table 3.

Case	Selection	Speciation	Replacement (5%)	Convergence (Fitness > 0.99?)	Mean MSE (20 runs)	Average Generational Cycle it takes to converged
1	SUS	No	No	No	N.A.	N.A.
2	SUS	No	Yes	Yes	0.0020	77.7
3	SUS	Yes	No	Yes	0.0063	54.5
4	SUS	Yes	Yes	Yes	0.0021	64.9

Table 3 Experiment to Study the Impact of Speciation and Replacement Strategy on the GA by Running it 20 Times for each Cases

From Table 3, it is observed that in Case 1, when SUS is implemented without speciation and replacement strategy, the solution does not converged. The training plot in Figure 17 shows the population fitness fluctuates considerably for this case.

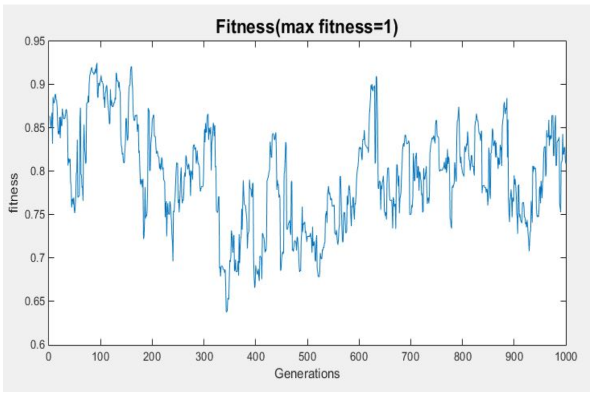


Figure 17 Without Speciation and Replacement Strategy

When the replacement strategy is implemented in Case 2, convergenced occurred with the MSE of 0.0020. This is comparable with the result for Case 4 where both speciation and replacement strategy are implemented. However, in Case 4, the average generational cycle of 64.9, where the solution converges, is shorter than that in Case 2 where the average generation cycle is 77.7. In case 3, although convergence occurred at an earlier average generational cycle of 54.5, the average MSE result was higher than in Case 2 or 4. The experiment showed that when both speciation and replacement strategy are present in the GA, the results were better than if speciation or replacement strategy were to be implemented separately.

5. Conclusion and Future Work

In this paper, we have proposed novel mechanisms of the GA to improve the convergence rate for NN modeling while preventing premature convergence. We have first assessed how in particular a new encoding scheme that embodies the NN weights and structure in a matrix representation can enhance the efficiency of the GA search. We have then further investigated the effects of sub-populations and a replacement strategy on the efficiency of the GA operations.

The results from the experiments are promising, which can be attributed to the following factors. First, the matrix encoding for the NN structure and weights, which is compact and sufficient to represent the possible solution space. Second, speciated compatible individuals, where individual competes locally instead of globally for survival. This approach protects potential individuals from early extinction and allows exploration to flourish for innovation. Third, a corresponding replacement strategy, which is presented to eliminate weakest individuals with top individuals, hence advancing the fitness of the overall population. Together, these mechanisms have enhanced the efficiency of the GA operations.

With the evolutionary NN of an optimal structure, the NN based non-linear black-box modeling now offers improved accuracy and dynamics in forecasting the energy consumption of a cloud data center. This could help data center owners eliminate under-provisioning or over-provisioning and manage resources scheduling more optimally [12].

For future development to improve the robustness of the solution, two areas have been identified for further investigation. First, how to ensure species diversity both in initial distributions of the subpopulations as well as during the candidate selection to discourage premature convergence. Second, in black-box modeling of non-linear systems, an apparent drawback is that the physical meanings of parameters, structures and their impacts on the system could not be reflected by the model. A ‘grey-box’ model combining the merits of the black-box model with physical principle based clear-box approaches can therefore be developed. This could lead to a model that would accurately predict data center energy consumption or cloud demand at the same time, giving data center owners the visibility of key factors affecting its performance and competitiveness.

References

[1] A.-A. Tantar, A. Q. Nguyen, P. Bouvry, B. Dorronsoro and E.-G. Talbi, *Computational Intelligence for Cloud Management Current Trends and Opportunities*, in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC),* 2013.

[2] I. S. Moreno and J. Xu, *Neural Network-Based Overallocation for Improved Energy-Efficiency in Real-Time Cloud Environments*, in *Proceedings of IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC),* 2012.

[3] J. Gao, *Machine Learning Applications for Data Center Optimization*, Research at Google, 2014.

[4] S. Pelley, D. Meisner, T. F. Wenisch and J.W. VanGilder, *Understanding and Abstracting Total Data Center Power,* in *Proceedings of the 2009 Workshop on Energy Efficiency Design (WEED),* 2009.

[5] J. Prevost, K. Nagothu, B. Kelley and M. Jamshidi, *Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models*, in *Proceedings of the IEEE 6th International Conference on System of Systems Engineering (SoSE),* 2011.

[6] Z. Song, B. T. Murray, B. Sammakia and S. Lu, *Multi-objective optimization of temperature distributions using Artificial Neural Networks*, in *Proceedings of the IEEE 13th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm),* 2012.

[7] J. Fulcher, *Computational Intelligence: A Compendium*, Springer-Verlag Berlin Heidelberg, 2008.

[8] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition*, Heaton Research, 2008.

[9] T. Pencheva, K. Atanassov and A. Shannon, *Modelling of a Stochastic Universal Sampling Selection Operator in Genetic Algorithms Using Generalized Nets,* in *Proceedings of the 10th International Workshop on Generalized Net,* 2009.

[10] W. Li., H. Yang, Z. Luan and D. Qian, *Energy Prediction for MapReduce Workloads*, Dependable, in *Proceedings of the IEEE Ninth International Conference on Autonomic and Secure Computing,* 2011.

[11] N. Al-Messabi, C. Goh, I. El-Amin and Y. Li, *Heuristically Enhanced Dynamic Neural Networks for Structurally Improving Photovoltaic Power Forecasting,* in *Proceedings of the IEEE 2014 International Joint Conference on Neural Networks (IJCNN),* 2014, pp. 2820–2825.

[12] Z. Zhan, X. Liu, Y. Gong, J. Zhang, S.H. Chung, and Y. Li., *Cloud Computing Resource Scheduling - a Survey of Its Evolutionary Methods*, ACM Computing Surveys, vol. 47, 2015.

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation, 2nd Edition,* Prentice-Hall International, Upper Saddle River, N.J, 1999.

[14] K. O. Stanley and R. Miikkulainen, *Evolving Neural Networks through Augmenting Topologies,* Evolutionary Computation, 10(2):99-127, 2002

[15] J.-P. Li, X.-D. Li and A. Wood, *Species Based Evolutionary Algorithms for Multimodal Optimization: A Brief Review*, in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC),* 2010, pp. 1-8