Evolutionary Neural Network Based Energy Consumption Forecast for Cloud Computing

Yong Wee Foo[1,2], Cindy Goh[1], Hong Chee Lim[1], Zhi-Hui Zhan[3], Yun Li[1]
[1]School of Engineering, University of Glasgow, Glasgow, U.K.
[2]School of Engineering, Nanyang Polytechnic, Singapore
[2]Foo_Yong_Wee@nyp.edu.sg
[3]Department of Computer Science, Sun Yat-sen University, Guangzhou, P. R. China
[3]zhanzhh@mail.sysu.edu.cn

***Abstract*—The success of Hadoop, an open-source framework for massively parallel and distributed computing, is expected to drive energy consumption of cloud data centers to new highs as service providers continue to add new infrastructure, services and capabilities to meet the market demands. While current research on data center airflow management, HVAC (Heating, Ventilation and Air Conditioning) system design, workload distribution and optimization, and energy efficient computing hardware and software are all contributing to improved energy efficiency, energy forecast in cloud computing remains a challenge. This paper reports an evolutionary computation based modeling and forecasting approach to this problem. In particular, an evolutionary neural network is developed and structurally optimized to forecast the energy load of a cloud data center. The results, both in terms of forecasting speed and accuracy, suggest that the evolutionary neural network approach to energy consumption forecasting for cloud computing is highly promising.**

Keywords—hadoop, cloud computing, energy efficiency, evolutionary computing, genetic algorithm, neural networks

I. INTRODUCTION

Social media, search engine and mobile computing all have one thing in common. They harness the abilities of cloud to analyze, store and share information. The expanding role of cloud computing in our everyday life have led to new energy challenges as service providers intensify the buildup of cloud data centers, facilities and IT infrastructure to support the demand for cloud computing. For example, Facebook's data center energy consumption surged by 33% in 2012 to 678 million kWh of energy from the year before. Its total data center energy consumption is up by another 16% in 2013 to 822 million kWh as the Internet Company massively scaled up its cloud infrastructure to meet strong customer demands [1]. The cloud computing energy demand will continue to face upward pressures driven by two emerging phenomena. Firstly, the Internet of Things (IoT) workloads are predicted to consume data center capacity by another 750% from 2014 to 2019 [2]. Secondly, high performance data analysis (HPDA) servers are predicted to grow robustly at a 23.5% compounded annual growth rate (CAGR) from 2013 to 2018, to $2.7 billion in 2018 [3]. These phenomena will add considerably to the already energy-intensive cloud data center. Therefore, critical action is needed to manage its energy efficiency.

The use of computational intelligence has been applied to solving complex real-world problems where conventional approaches such as clear-box modeling using physical principles or statistics are not feasible. When domain knowledge and physical meaning and interpretation about the system are available, clear-box approaches are sufficiently accurate for modeling system behaviour. However, in the presence of noise, nonlinearities and complexities associated with the system dynamics, computational intelligence approaches such as artificial neural networks (ANN) and evolutionary computation (EC), are more widely used to model the behaviour of such systems. In [4], Tantar et al. calls for cloud management solutions that combine queueing theory, optimization and machine learning due to the stochastic nature of cloud. In this light, we are particularly interested in the direction of evolutionary and nature inspired paradigms to solving today's energy efficiency challenges in cloud computing [5] [6].

In this paper, we develop an evolutionary neural network as a structurally optimal black-box model to forecast energy consumption of a dynamic cloud data center. Our approach includes several novel mechanisms of a genetic algorithm (GA), such as a structurally-inclusive matrix encoding and species parallelism, which help maintain an overall increasing fitness to overcome slow convergence. The model is trained using data from a set of MapReduce jobs obtained on a 120-core Hadoop cluster and validated using unseen data. Results obtained show promising improvements to both forecasting accuracy and speed of energy consumption in the cloud data center.

Hadoop [7] is chosen as the testbed as it presents an interesting cloud computing model [8] [11] [14]. Hadoop is an open source framework that uses data and task replication to achieve data parallelism. Its architecture enables users to create large-scale massively parallel distributed processing system at reasonable cost. Hadoop consists of two layers, namely the Hadoop Distributed File System (HDFS) [9] layer and the MapReduce [10] layer. Hadoop uses the HDFS layer to partition and replicate datasets across multiple nodes for load balancing and fault tolerance. The MapReduce layer

then take advantage of the distributed data and the large number of nodes for concurrent processing to improve application performance.

In Section II, related works in energy efficiency in big data cloud are reviewed. Section III describes the details of our novel evolutionary neural modeling approach. In Section IV, the experimental setup is explained in more detail. Results and discussions are presented in Section V. Section VII concludes the paper with recommendations for future work.

II. Related Work

Related work in the area of energy efficiency for Hadoop cloud is discussed below and summarized in Table 1.

A. Energy and Thermal-Aware Workload Scheduling

By design, Hadoop MapReduce schedulers are not energy-aware, that is, the scheduling of workload does not consider its effect on data center energy consumption or the data center thermal environment. For instant, Hadoop's First-In-First-Out (FIFO) scheduler simply assigns the tasks to the available servers based on the job arrival sequence. Significant research has been developed around thermal- and energy-aware scheduling. Kulkarni [16] proposed to schedule tasks based on thermal model in data center to minimize the heat dissipated by the nodes and subsequently the cooling power overheads. Krish [15] proposed a heterogeneity-aware and power-conserving task scheduler for Hadoop workload scheduling algorithm that allocates workloads to servers based on statistical analysis to discover application-resource match for optimal energy performance that meets SLA.

B. Turning off Idle Nodes

Several works have been built on Barroso's concept of energy-proportional operation [18]. For instance, Leverich [17] proposed to turn off nodes to save energy in a Hadoop cluster by aggregating "live" data into subset nodes of the cluster and turn off nodes outside the subset. In [19], Amur presented Rabbit, a power-proportional distributed file system that takes advantage of systems like HDFS to achieve power performance efficiency. The concept lies in its data layout arrangement such that a minimal number of powered-up nodes are active. The primary and secondary dataset replica made available to the cluster ensures minimal nodes are powered up during node failure. Lin [20] proposed an energy efficient and resilient data layout policy called eStor, where some replicas are stored in the sequential way, while the other are placed in the random way. This solution addressed both the problems of data availability and rebuild parallelism without disrupting data popularity when turning off nodes.

C. Improving Storage Efficiency

In the area of storage efficiency, Fan [21] suggested lowering the overheads caused by replication by introducing erasure coding RAID in Hadoop HDFS. Cheng [22] examined a novel way to get better performance and higher disk utilization, through the replication policy of HDFS. The author demonstrated an Elastic Replication Management System (EMRS) for HDFS which is elastic and adaptive to data popularity. Wei [23] proposed the use of Low Density Parity Check (LDPC) coding which further improves the storage efficiency of HDFS RAID.

D. GreenHDFS

Kaushik [24] proposed a GreenHDFS (Green Hadoop Distributed File System) concept that logically divides the Hadoop cluster into hot and cold zones. Datanodes in the hot zone contain active data or data that are frequently accessed, whereas datanodes in the cold zone contains inactive data or data that is not frequently accessed. Relying on insightful data-classification and energy-conserving placement policy, this method attempts to guarantee, substantially long periods (several days) of idleness in a significant subset of servers in the cold zone. The Hadoop cluster is scaled down to allow servers in the cold zone to transit to an inactive, low power consuming sleep/standby state to achieve energy savings. However, it suffers energy and performance penalty in "waking" the servers especially if the frequency of server awakening is high. To compensate this drawback, Kaushik in a later paper [25], presented the predictive GreenHDFS where a predictor is used to learn the associations between the directory hierarchies and file attributions. The predictive model then guides policies to improve the energy-performance tradeoff.

III. Evolutionary Neural Networks

Evolutionary computing (EC) is a nature inspired computation algorithm that imitates the evolutionary mechanism such as selection and reproduction, with the objective to automatically create, combine, and select data structure to find solutions to optimization problem [37]. Evolutionary neural network (ENN) is a type of machine learning that uses the EC approach to train an artificial neural network to find the optimized structure to a problem. Therefore, ENNs consist of two main components – the ANN and the GA.

A. Artificial Neural Networks

ANNs [29] are computational models that mimic how a human brain learns. It has been used extensively in pattern recognition like facial and handwriting recognition, as well as regression analysis and for function approximation [38]. The function approximation property of NN is based on the universal approximation theorem [26], which summarily states that simple NN, such as the multilayer feedforward NN architecture, can represent a wide variety of interesting functions when given appropriate parameters. This property makes NN modeling a suitable and adequate approach to approximate unknown non-linearities, with it complexity and diversity, of a dynamic cloud data center. In order for the NN model to be deployed, it must undergo a training process, where the NN weights and structure are continually updated using data from the actual system to arrive at a

model most closely represent the system under test. The most common training method use is the backpropagation (BP) algorithm, a gradient descent method [28] that calculates the gradient of a cost function with respect to all the weights in the network.

In the BP algorithm, the weights are adjusted at each iteration to minimize the cost function until a solution is reached. However, there is an issue with this method as the final solution tends to converge on the local optima [27] instead of the global optima. One of the solutions to this problem is to use an evolutionary algorithm like the genetic algorithm, which searches the solution space for the global optima.

Figure 1 shows an artificial neuron also known as a perceptron. The perceptron is the simplest form of the neural network that contains only a single layer with a single node. The output of the single perceptron, Y is determined by the weighted sum of the inputs and applying an activation function on the sum. It can be mathematically expressed in (1):

$$Y = \varphi(w_2 * x_1 + w_3 * x_2 + w_4 * x_3 + w_1 * 1) \qquad (1)$$

where,

Y is the output of the perceptron
x_i is the value at input i
w_i is the connection weight at input i
φ is the activation function

The activation function used is the sigmoid function given in (2) below:

$$\varphi(Y) = \frac{1}{1+e^{-Y}} \qquad (2)$$

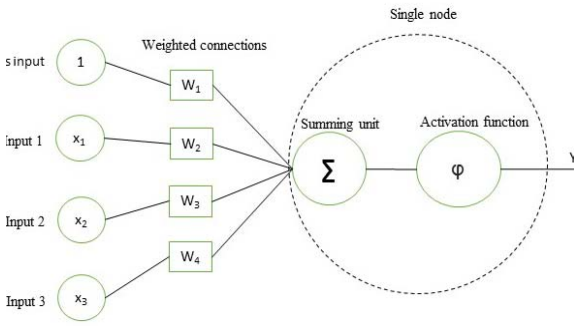


Figure 1. A Single Perceptron

In our experiment, we use the feedforward multi-layer perceptron (MLP) consisting of one input layer, one hidden layer and one output layer. Based on the universal approximation theorem, any arbitrary continuous function can be arbitrary well approximated by an MLP. Our network has a minimum of 2 and a maximum of 21 hidden neurons.

B. Genetic Algorithm

Genetic algorithm [30] is a search optimization technique that models after the Darwinian observation of natural selection. The genetic algorithm works by evolving a set of solutions to create the next generation that possesses an improved overall fitness towards the solution. Hence, it has the ability to achieve global optimization unlike other optimization algorithms like the backpropagation.

Genetic algorithm starts by initializing a population of individuals. Each individual represent a possible solution in the solution space. A cost function is selected to evaluate the fitness of each of the individuals. The initial population is subjected to the process of evolution where genetic operators like crossover and mutation is used to create a whole new generation of individuals. During the process, less healthy individuals have a lesser chances of participating in the crossover and mutation thereby ensuring that the genetic information of healthier individuals are passed on to the next generation. The two main considerations in genetic algorithm are the encoding of the chromosome and the genetic operators which are used to evolve the set of chromosomes.

Chromosome are actual structure containing the gene. The genotype is the abstract collection of genes possessed by an individual [31]. The gene has a value and a position in the genotype. There exist in nature, a mapping between the genotype and the phenotype, the latter are properties of the organism. In using GA to solve various optimization problems, the problem to be solved (phenotype) needs to be represented in its genotype equivalent. This representation of the problem into its genotype form, is known as encoding. In some application, determining the best encoding can be a somewhat complex issue. It is also possible that a poor encoding scheme may lead to poor solutions. The degree of encoding is key [31]. There is thus a challenge to devise an encoding scheme with genotypic building blocks simplified for GA implementation (e.g. initialization, selection and crossover) while at the same time, encapsulates as much details into the genotype as possible to provide a good coverage of the solution space. A building block is a solution component that can be split, recombined or reshuffled with other building blocks in a constructive way that produces healthier offspring from which the final optimal solution is found.

1) Encoding Scheme

There are largely two encoding scheme – direct and indirect. In direct encoding, NN structure (phenotype) such as weight values, connection information, etc. are encoded into the genome. This is in contrast to indirect encoding, whereas rules, which carry structural information of the NN, are encoded. In our approach, direct encoding is used where the genotype is map directly from the phenotype. The neural network weight values and connection topologies are directly represented in the chromosomes.

TABLE 1. RELATED WORK SUMMARIZED IN RESPECTIVE CATEGORIES

Author/Title (First author)	Category	Key Concept	Method/Approach	Experiment/Results	Remarks
K. Krish/ Towards Energy Awareness in Hadoop [13]	Hardware heterogeneity aware workflow scheduling.	Uses statistical analysis is performed to determine suitable resource-application match.	Profiles the performance and the energy characteristics of Hadoop applications on different hardware platforms. Uses metrics such as CPU, memory, storage and network usage to improve energy efficiency by matching current and future application to the resources while ensuring performance is not compromised.	Simulation results of a 300 node cluster using publicly available Facebook production traces as inputs show that time improves by 12.8%, while using 21% less power.	The frequent replacement or upgrades due to hardware failure is a common occurrence in a large-scale cluster, would post a scalability challenge for this approach.
S. Kulkarni /Cooling Hadoop: Temperature Aware Schedulers in Data Centers [14]	Energy, thermal-aware job scheduling.	Measures CPU and Disk utilization and temperature and maintains a predefine threshold temperature across nodes by load balancing and assigning tasks to the coolest nodes first.	Two thermal-aware schedulers were designed to balance out the temperature and reduce power consumption cost in the data centre. Dynamic scheduler schedules job based on CPU and disk temperatures and utilization feedback given by the slave nodes. A second static scheduler then assigns the job to the slave nodes as well as to store the profile of the slave nodes. The new schedulers are implemented as a layer above Hadoop's FIFO.	The new schedulers on top of Hadoop's FIFO has improved 10-15% of energy cost.	The method saves power but comes at the cost of performance. Execution time can increase as high as 70%.
J. Leverich/On the Energy (In)efficiency of Hadoop Clusters [15]	Energy-aware replication invariant. Power-proportionality.	Covering subset ensures minimum availability of all request data while the rest can be powered down to conserve energy	The proposed covering subset invariant serves two purposes. First, to ensure immediate availability of data and second, to allow all nodes not in the covering subset to be gracefully power down to save energy.	Energy consumption saving achieved between 9% and 50%	It could leave the cluster in a vulnerable state for when one of the "live" node is down, there will be penalty for both application and energy performance from "awaking" the nodes.
H. Amur/ Robust and Flexible Power-Proportional Storage [18]	Power-aware data layout policy. Power-proportionality.	Equal-work policy for even division of workload across clusters to achieve power-proportionality	The concept of Rabbit is to carefully arrange an equal-work data-layout to provide power-proportionality. Secondary replicas of blocks ensure minimal disruption of power-proportionality when nodes fail.	Experiment results on 100 nodes with 10k blocks, using 5 primary nodes and 4 replicas, shows that Rabbit can provide ideal power-proportionality and full data availability across a range of settings whereas low as 5% of the nodes are active if 4 replicas of the data are stored.	While a smaller primary set achieves greater potential energy savings but also leads to write bottleneck. A write-offload solution solves the problem but at the expense of management overheads. Low data rebuild parallelism.
Lin/estor Energy Efficient and Resilient Data Center Storage [19]	Power-aware data layout policy. Power-proportionality	eStor store some replicas in the sequential way, while the other are placed in the random way.	During low I/O utilization period, turn off up to (m -1)/ m of nodes (m is the number of replicas that are placed in the sequential way). Analyze the data rebuild parallelism of eStor, the number of nodes the lost replicas can reside on when a server failed. Consider data popularity policy to maintain more replicas for hot data when turning off servers.	Experiments with eStor implemented in Hadoop demonstrate a 40% energy savings and have high data rebuild rate.	In a large-scale cluster and depending on the fraction of replica placed sequentially, data replication may tend toward random data-layout for the portion under random placement policy.
Fan/ DiskReduce: RAID for Data-Intensive Scalable Computing [20]	Improvement of Hadoop HDFS. Erasure coding RAID.	Replace HDFS replication of 3, with RAID 5 or RAID 6 to improve spatial and energy efficiency.	The approach first combines RAID 5 and mirroring encoding with RAID 6 encoding to lower the overhead of triplicated down to RAID-class redundancy. It then delays encoding by an hour so accesses to the 3 copies remains during the hour.	Results show a spatial overhead savings of 70% using RAID 6 over triplication without performance degradation.	Approach is asynchronous hence encoding can be delayed for a prolonged period if "hot" files are continually accessed.

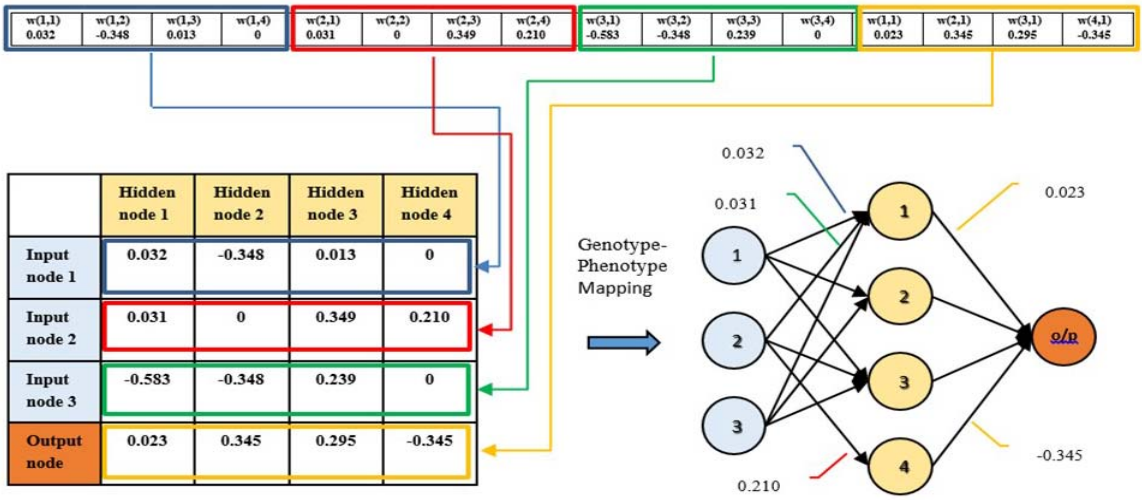
Author/Title (First author)	Category	Key Concept	Method/Approach	Experiment/Results	Remarks
Cheng/ERMS: An Elastic Replication Management System for HDFS [21]	Data-placement strategy. Erasure coding RAID.	Use of complex event processing (CEP) engine to analyze HDFS audit logs to distinguish real-time data types. Place data into active/standby storage for HDFS.	Implements replica placement strategy whereby ERMS dynamically increases extra replicas for hot data and performs cleaning up of these extra replicas when the data cool down. Uses erasure codes RAID for cold data and triplication for normal data. Achieve energy savings storage efficiency.	The result shows that ERMS effectively improves the reliability and performance of HDFS and reduce storage overhead and energy consumption.	ERMS CEP engine has to operate on real-time or near real-time mode for the scheduling and data placement approach to be effective.
Wei/ Reliable and efficient distributed storage with LDPC code **[22]**	Improvement of Hadoop HDFS. Erasure coding LDPC.	Use of LDPC coding to achieve the best trade-off between storage efficiency and reliability	Integrate LDPC coding with Hadoop HDFS. Uses an optimized equation-based repair algorithm to minimize the computational complexity and repair traffic and improve storage efficiency.	Test results show LDPC code can tolerate any 5 erasures with redundancy rate only 2.6 compared with 6-replication method. This translates to 56.7% space saving.	LDPC coding is a promising alternatives to standard erasure coding such as Reed-Solomon.
R. Kaushik/ Evaluation and Analysis of GreenHDFS : A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System [23]	Data classification and placement. Power-proportionality.	Focus on data-classification techniques on Hadoop logs and metadata files using evolution and lifespan analysis to extract energy savings by doing energy-aware placement of data.	Hadoop server clustering into Hot and Cold Zones and relies on insightful data-classification driven energy-conserving data placement to realize guaranteed, substantially long periods (several days) of idleness in a significant subset of servers in the Cold Zone. Scale-down servers by manufacturing idleness then migrating workloads and their corresponding state to fewer machines during periods of low activity.	Using production data from Yahoo! Hadoop cluster of 2600 servers with a dataset size of 6 Petabytes, the experiment results show that GreenHDFS is capable of achieving 26% energy savings.	HDFS distributes data chunks and replicas across servers for resiliency, performance, load-balancing and datalocality reasons. Such data placement makes it hard to generate significant periods of idleness in the Hadoop clusters.
R. Kaushik/ Predictive Data and Energy Management in GreenHDFS [24]	Prediction of temperature of the file being accessed. Power-proportionality	Ridge Regression is used to discover the statistical correlation between the directory hierarchies and file attributes.	Using the strong statistical corelations between the file attributes and the hierarchical directory structure from which the files are organized to a) predict and decide an initial placement of a file in the Hot or Cold Zone, b) foretell when a file becomes cold, c) proactively create, move or delete files based on the file heat predictions.	Results show that predictive GreenHDFS experienced higher energy savings, less performance degradation and higher free storage space savings in the Hot zone compared to reactive GreenHDFS.	Aggressive energy savings algorithm, results in high number of file access to the Cold Zone which degrading performance due to the time needed to transition the server from idle or off mode to active mode.

Our unique GA representation scheme has the advantage of embodying the NN weights and connection characteristics into a chromosome that could be easily transformed into a matrix that enhance the GA operations.

To illustrate our encoding scheme, Figure 2 shows the chromosome encoding with information of a NN with 3 input nodes, 4 hidden nodes and 1 output node. The first 12 genes represent the connection between input node *i* to hidden node *j* and is denoted by *w(i,j)*. The last 4 genes represent the connection between the hidden node *j* to output node 1 and is denoted by *w(j,1)*. A value of '0' in the gene indicates the absence of a connection whereas a non-zero value indicates the presence of a connection. The values in the matrix are the connection weights. This encoded chromosome can be easily expressed in a 4 x 4 matrix as shown in Figure 4. Figure 3 shows the complete genotype (Figure 3(a)) to phenotype (Figure 3(b)) mapping.

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



3 (a) Genotype

3 (b) Corresponding NN (Phenotype)

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

To further illustrate our encoding scheme, we define a 4x4 matrix in Figure 4. The matrix nn_ind represents the individual NN structure in Fig 3(b). The first 3 rows of the matrix denote the input node connections to the hidden nodes. The last row denotes the output node connections to the hidden nodes. For instance, nn_ind(1,1), nn_ind(1, 2) and nn_ind(1, 3) indicates the presence of 3 connections from input node 1 to the hidden nodes 1 to 3 with the weight values of 0.032, -0.348 and 0.013 respectively. Likewise, nn_ind(4, 1), nn_ind(4, 2), nn_ind(4, 3) and nn_ind(4, 4) indicates the presence of 4 connections from the output node to the hidden nodes 1 to 4 with the weight values of 0.023, 0.345, 0.295 and -0.345 respectively. Notice that nn_ind(1, 4), nn_ind(2, 2) and nn_ind(3, 4) all have a weight value of '0'. This indicates that there are no connections between input node 1 to hidden node 4, input node 2 to hidden node 2 and input node 3 to hidden node 4. And since all elements in the last row contain a weight value, this matrix represents a NN structure with 4 hidden nodes.

$$\text{nn\_ind} = \begin{bmatrix} 0.032 & -0.348 & 0.013 & 0 \\ 0.031 & 0 & 0.349 & 0.210 \\ -0.583 & -0.348 & 0.239 & 0 \\ 0.023 & 0.345 & 0.295 & -0.345 \end{bmatrix}$$

Figure 4. A 4x4 Matrix Representation of the NN structure in Fig. 3(b)

2) Fitness Function

Without loss of generality and for the ease of analogy with conventional optimisaiton, the fitness function for the GA is the cost function J_{MSE} which is the Mean Square Error (MSE) or the average squared difference between the actual output, *a* and target, *t* of the neural network as shown in (3):

$$J_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(t_i - a_i)^2 \quad (3)$$

This output *a* (Eq. 6), of the NN is calculated from the following steps with the information from the connection matrix and the input from the training dataset.

$$\overline{s} = [inputs] * [\overline{w_{ih}}] \quad (4)$$

$$\overline{h} = \frac{1}{1+e^{-\overline{s}}} \quad (5)$$

$$a = \sum \overline{h} * \overline{w_{ho}} \quad (6)$$

where,

$\overline{w_{ih}}$ is the vector of connection weights between input and hidden nodes.

$\overline{w_{ho}}$ is the vector of connection weights between hidden nodes and output node.

$\overline{h}$ is the activation function

3) The Genetic Algorithm Operation

The genetic algorithm operation consists of mainly the process of selection, reproduction and mutation whereby fitter individuals dominates and reproduces while the weaker ones die out, simulating the biological world of natural selection and survival of the fittest.

a) Initialization: The genetic algorithm flow is shown in Figure 6 and its corresponding psuedo code in Table 2. The evolutionary process begins with the random initialization of individuals. At the start of each genetic cycle, the fitness of the individuals is calculated using (3) and ranked accordingly. A "kill" operator is implemented to replace the lowest ranking individuals with the top. The "kill-percentage", set at 5% of population, determines the amount of individuals that will be replaced by the top individuals. The replacement strategy keeps the average population fitness on a progressive course by purging of the weaklings.

b) Speciation: In the algorithm, we adopt the concept of speciation [32]. Speciation divides the population into sub-populations to protect and grant time for individuals to evolve to healthier organisms without the threat of any one species taking over the entire population too rapidly. This way, premature extinction is withheld thereby preserving structural innovation and giving potential solutions within the species, a chance to thrive. Using the number of hidden nodes as the compatibility function, individuals according to their species are subsequently categorized. In our GA, we set the number of species to to 20.

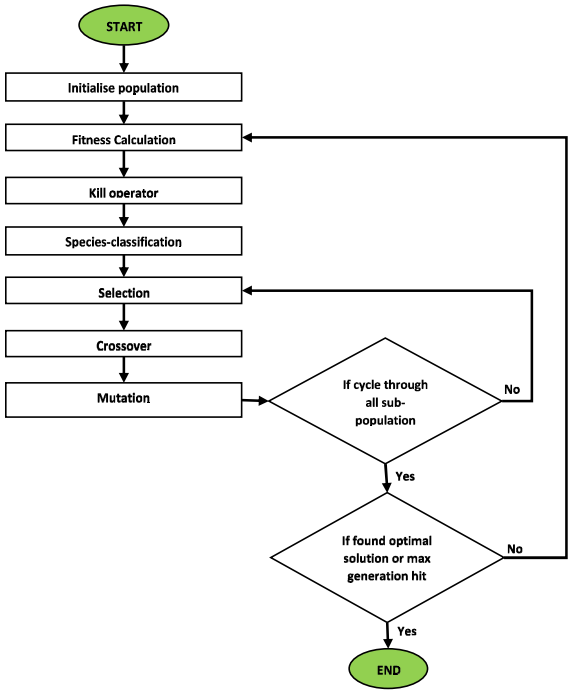


Figure 6. Genetic Algorithm Flow

c) Selection: After the speciation process, the selection process begins in the preparation for crossover.

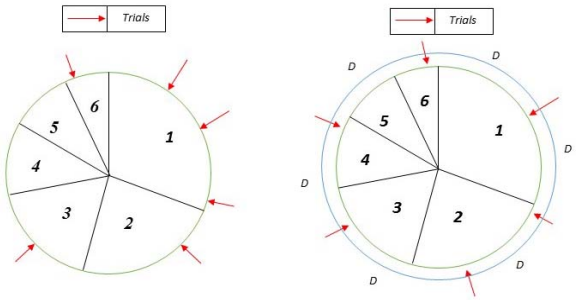


Figure 5(a). Ranked RWS Figure 5(b). SUS

Figure 5. Comparison between RWS on SUS

One of the more common selection algorithm used is the Roulette Wheel Selection (RWS) [34]. In roulette selection every individual is assigned to a section of an imaginary roulette wheel according to their fitness with the fittest individual having the biggest section of the wheel as shown in Figure 5(a). A random number (pointer) is generated and the individual whose section spans the position of the pointer is selected. One shortcoming with this selection scheme is that if the fitness of the fittest individual is very large, it will dominate majority of the roulette wheel causing the selection process to be over bias towards that individual. One solution to this problem is to implement a ranked-base roulette selection by assigning the section of the roulette wheel according to the fitness rank. This will ensure that there is no bias towards the fittest individual but it does not guarantee minimal spread which is the range in the possible number of times each individual is selected. In our work, we use the Stochastic Universal Sampling (SUS) selection scheme, which in constrast, is non-bias and ensures a minimum spread is maintained [33]. 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 allowing the fittest individuals to dominate the candidate space prematurely. To illustrate the point, Figure 5(a) shows the RWS for 6 individuals out of a population of 6. Here, 6 random number is generated and the 6 selected individuals are,

Selected individual => [1 1 1 2 3 6]

In Figure 5(b), SUS in this case has only 1 random number generated with the rest of the trials equally spaced after the random trials. The distance between the trials can be calculated using (7). This method will ensure that there is no bias and guarantee minimal spread.

$$D = \frac{1}{total\ number\ of\ trials} \qquad (7)$$

The selected individuals using the SUS are,

Selected individual => [1 2 2 3 4 6]

d) Crossover: Crossover is one of the main genetic operators use to evolve the chromosome. The process of crossover takes two or more parents' chromosome to reproduce one or more child. The motivation behind the crossover is that the child will inherited the best partial solution from the parents. In our experiment, we use two different crossover techniques, one for intra-species crossover, and the other for inter-species crossover. Intra-species crossover is a phenotype crossover. In intra-species crossover, we apply a single-point crossover at the mid-point of the hidden layer in the NN structure, where the parents' genetic material 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. This safeguard is to protect structural innovation and prevent the threat of any one species from taking over the entire population. Whereas in the inter-species crossover, it is a genotype crossover whereby we apply a single-point crossover at the mid-point of the genotype between two parents of different species (see Figure 8). 90% of the population will experience intra-species crossover and 1% will experience inter-species crossover.

TABLE 2. GA PSEUDO CODE

```
Genetic Algorithm ()
[
Set control parameters;
Initialise population;
do while stop conditions not met
  [
  Calculate fitness;
  Replace weak individuals;
  Classify species;
  do while all species are not cycled through
     [
     select solution for next generation
     perform crossover
     perform mutation
     ]
  end while
]
end while
]
End
```

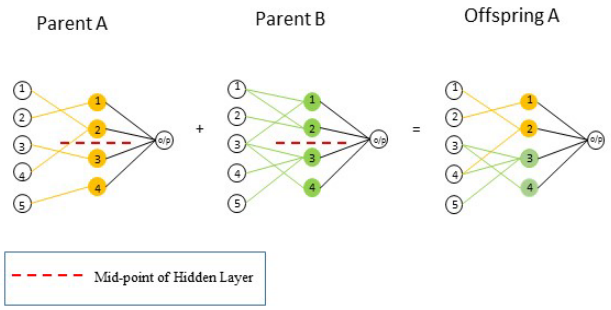


Figure 7. Intra-species Phenotype Crossover of 4-hidden neuron Sub-population

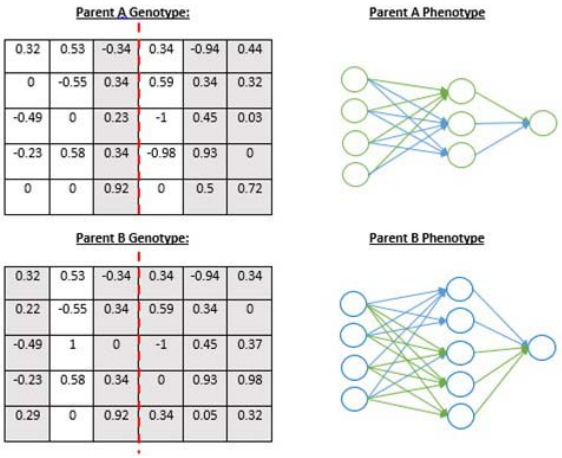


Figure 8. Parents before Inter-species Crossover

In inter-species crossover, the resulting child is not guaranteed to be in the same species as the parents (see Figure 9). The recombination at the mid-point of the genotype may produces offspring of outside of the parents' species which allow new search spaces to be explored.

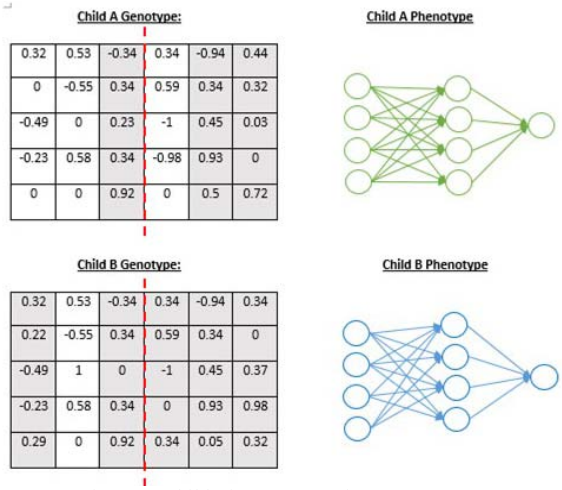


Figure 9. Child after Inter-species Crossover

e) Mutation: Mutation is a secondary operator with the role of restoring lost genetic material. We set a small percentage of individuals in the population to be selected for structural and weight mutation. Structural mutation are introduced to potentially change the genetic makeup of the individual from one species to another to explore other search spaces. The weights mutation is introduce to aviod the solution from being trapped in a local minima.

f) Stop condition: To terminate the GA execution, a stop condition is specified. The execution is stopped after 100 generation has passed or the predefine high fitness individual is found or after all the the individuals in the population has attained a certain amount of homogeneity, that is the overall fitness of the population does not change much.

The control parameters for the GA operation and their respective values are shown in Table 3.

TABLE 3. GA CONTROL PARAMETERS AND VALUES

GA 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.	0.9
xo_rate_inter	The inter-species crossover rate.	0.01
mu_rate_wt	The mutation rate for the connection weights.	0.015
mu_range_wt	The range of values the mutated weights will take on.	-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.	0.01
kill_percentage	The kill percentage number.	0.05

IV. Experimental Setup

A. Hadoop Testbed

The testbed is a 120-core Hadoop cluster set up to perform the experiments. Energy related matrices from running Hadoop MapReduce jobs are extracted and the data collected are used to train and calibrate the models. The Hadoop testbed comprises of one Hadoop namenode and four Hadoop datanodes. The namenode is running on a HP ProLiant DL360P Gen8 server that comes with 64 GB memory, dual socket Intel(R) Xeon(R) CPU E5-2667 @ 2.90GHz with a 6-core CPU and hyper-threading technology. The datanodes is running on HP ProLiant DL380P Gen8 server that comes with 48 GB memory, dual socket Intel(R) Xeon(R) CPU E5-2640 @ 2.50GHz with a 6-core CPU and hyper-threading technology. With the hyper-threading aware operating system, the Hadoop cluster of 5 nodes consists of a total of 5x2x6x2 or 120 logical core.

All nodes are installed with CentOS 6.5 operating system with Facebook Apache Hadoop version 0.20.1. The nodes are running on bare-metal hardware and are interconnected via a top-or-rack (TOR) Gigabit Ethernet switch with 1 Gigabit per second link to all the nodes. Figure 10 shows the physical infrastructure and connectivity of the Hadoop testbed. The Hadoop software configuration parameters are tuned as shown in Table 4.

Table 4. Hadoop Parameters and Values

Hadoop Parameter	Value
Maximum map task	6
Maximum reduce task	4
Replication factor	2
mapred.child.java.opts	512MB
io.sort.mb	200MB
io.sort.spill.percent	0.9
io.sort.factor	20
dfs.block.size	128MB
MapReduce Jobs	**Number**
WordCount	22
Sort	23

B. Energy Related Metrics

A total of 12 input metrics were identify and collected. In identifying the metrics for energy consumption of MapReduce workloads, we considered the work in [35]. In a typical MapReduce job, there exists three phases – The Map phase, the Shuffle phase and the Reduce phase. The computation of Map and Reduce functions are performed by the CPU calculation. The metric - number of instructions, (the sum of Map and Reduce tasks) is representative of the complexity of the MapReduce job. This metric captures the energy consumed by the CPU.

The read and write of the Map and Reduce functions are performed by the disk IO. Data are read and write to and from the HDFS storage during this phase. Hence the metrics - Map read/write and Reduce read/write in bytes, indicates the energy consumed by the disks.

The shuffle function generates data transmission where mapped data are moved (or shuffled) to the nodes where the Reduce function is performed on the data. As data

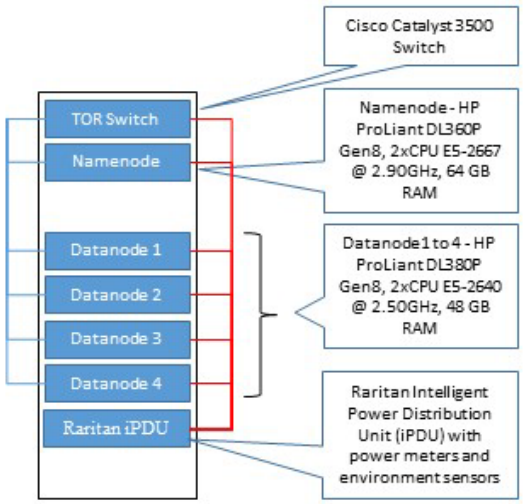


Figure 10. The Hadoop Testbed for the Experiment

is being transmitted over the network, this consumes energy as data needs to be preprocessed before sending and receiving at the end points as well as in transition at the intermediate network devices such as switches and routers.

Collectively, these metrics are monitored, measured and recorded from runtime MapReduce job counters which form the dataset to train the prediction model. The neural network input features are listed in Table 5.

TABLE 5. NEURAL NETWORK FEATURES

1.	No. of Map and Reduce Instructions	7.	Map Write (Gigabyte)
2.	CPU Utilization (%)	8.	Reduce Write (Gigabyte)
3.	System Load (%)	9.	Reduce Shuffle (Gigabyte)
4.	Memory Use (%)	10.	Network Bandwidth (Gigabit per second)
5.	Map Read (Gigabyte)	11.	File size (Gigabyte)
6.	Reduce Read (Gigabyte)	12.	Job Completion Duration (Hour)

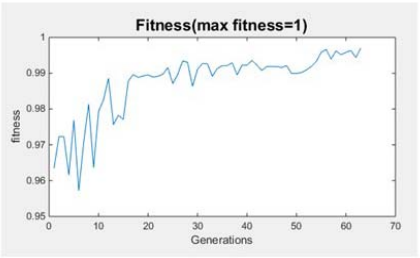
The output is the energy consumption measured in kilowatt-hour. Table 6 describes the training dataset for NN input and output, along with their method of collection.

V. RESULTS AND CONCLUSION

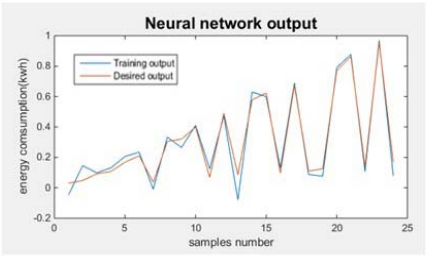
Figure 11 shows the result of the GA run. Fig 12(a) shows the fitness plot of the GA. The fittest individual has a fitness of 0.991 found in the 63th generation. Fig 12(b) shows the output plot. As it can be seen in this figure, the output of the actual data tracked the desired data plot indicating the network is learning to model after the actual system. Figure 13 shows the solution structure of the fittest individual optimized by GA. The network structure has a hidden layer of 9 neurons. We test the solution with dataset not used by the training and validation process. Figure 12(c) shows the test plot by the NN solution with an MSE of 0.096.

Figure 11 shows the sub-population distribution at various generations. At initialization (Figure 11(a)), the population shows a normal distribution. The demographic of the population changes after every generation. Here, we show a snapshot at the 20th (Figure 11(b)) and 63rd generation (Figure 11(c)). It can be seen that as the search progresses, new species were formed while weaker species were died out. The sub-population sizes for each species also change at each generation. At the 63rd generation, the solution converged.

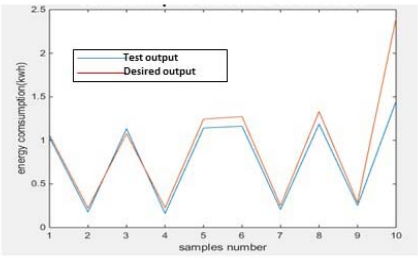
To further analyze the implementation of speciation in GA, we run the GA again to compare the solution fitness between speciation and non-speciation population. This time we modified the stop condition in the algorithm to allow GA to execute up to 100 generations. We plot the solution fitness taking the fittest individual of every generation and compare their results in Figure 14. The plot shows that for speciated population the solutions are fitter than those of the non-speciated population. This occurs for most part up to the first 70 generations. However, after 70 generations, the solution fitness for speciated population starts to plateau and even decline. This indicates that convergence has taken place and further execution of the GA would no longer help improve the solution. As for the non-speciated population, it took at least 70 generations to reach a good solution benchmarked by the fitness of the speciated population.



12(a). Fitness Plot

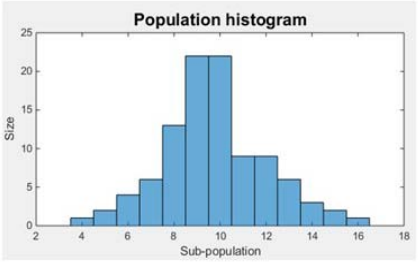


12(b). Training and Validation Plot

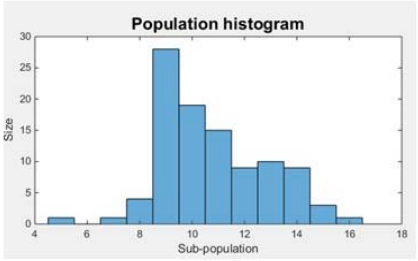


12(c). Test Plot

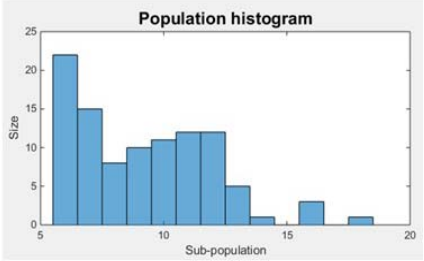
Figure 12. Plots Showing the Results of the GA



11(a) Initial distribution



11(b) Distribution after 20 Generations



11(c) Distribution after 63 Generations

Figure 11. Population Distribution

TABLE 6. ENERGY RELATED METRICS FOR HADOOP CLUSTER

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 resources spent on 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	GB GB GB GB	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	GB	Data transferred from Map to Reduce	Hadoop built-in counters
		10. Network Bandwidth	Gbps	Data transmitted and received	Ganglia
	Job Profile	11. File size	GB	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

To verify the results, we performed 20 runs for each case. Table 7 shows their average convergence rate. The results in Table 7 indicate that average convergence takes 64.9 and 77.7 generations for speciated and non-speciated population respectively.

TABLE 7. CONVERGENCE RATE FOR SPECIATED AND NON-SPECIATED POPULATIONS

Case	Speciation	Convergence	Mean MSE (20 runs)	Average Generations to convergence
1	No	Yes	0.0020	77.7
2	Yes	Yes	0.0021	64.9

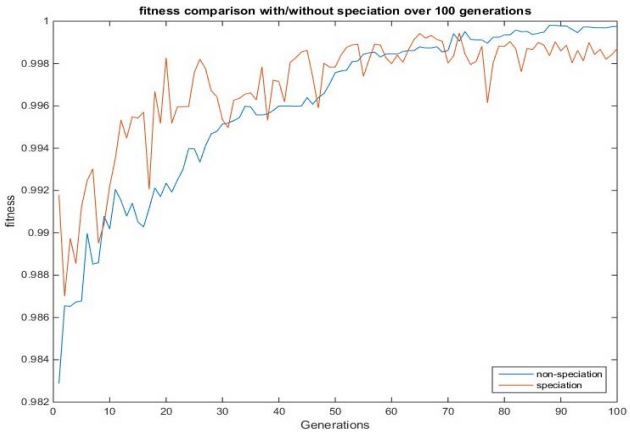


Figure 14. Speciation versus Non-speciation

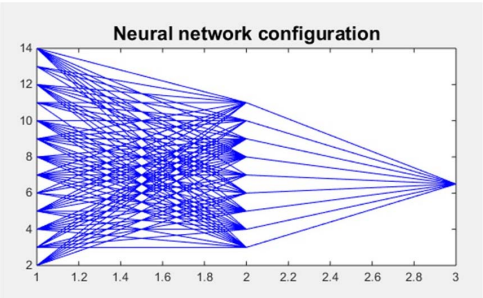


Figure 13. Solution of the NN Structure

VI. CONCUSIONS AND FUTURE WORK

In this paper, we have presented an ENN that can be structurally optimized to forecast cloud computing energy consumption. It is shown how 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 conducted further investigations on the effects of sub-populations on the efficiency of the GA operations.

The results from the experiments are promising, which can be attributed to the following factors. Firstly, the matrix encoding for the NN structure and weights, which is compact and sufficient to represent the possible solution space. Secondly, the implementation of speciation using number of hidden nodes as compatibility function, and using intra-and-inter species crossover and structural mutation operations, allows for the algorithm to explore other search spaces for global optima with good convergence rate. Together, these mechanisms have enhanced the efficiency of the optimization.

With the ENN, the non-linear black-box modeling approach can now offer improved accuracy and dynamics in forecasting the energy consumption of a cloud data center. This would help data center owners eliminate under-provisioning or over-provisioning and manage resources scheduling more optimally [36].

For future developments, we plan to take advantages of both black-box and clear-box models to develop a 'grey box' that will preserve the physical significance of the system parameters while at the same time model the nonlinear complexities of the cloud data center. An advantage of this approach is that key factors affecting the energy performance can be accurately extracted, allowing for a better understanding of the system.

REFERENCES

[1] Carbon and energy impact, https://www.fb-carbon.com/pdf/download2013.pdf, 2013.

[2] R. Villars, J. Koppy and C. MacGillvrary, "Impact of Internet of Things on Datacenter Demand and Operations," IDC, Doc # 255397, April 2015.

[3] S. Conway, C. Dekate and E. Joseph , "Worldwide high-performance data analysis 2014–2018 forecast," IDC, Doc # 248789, May 2014.

[4] A.-A. Tantar, A. Nguyen, P. Bouvry, B. Dorronsoro and El-G. Talbi, "Computational Intelligence for Cloud Management Current Trends and Opportunities ," IEEE Congress on Evolutionary Computation, 2013, pp. 1286-1293.

[5] J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," 6th International Conference onSystem of Systems Engineering (SoSE), 2011, pp. 276 –281.

[6] Y. Chang-tian and Y. Jiong, "Energy-aware genetic algorithms for task scheduling in cloud computing," Seventh ChinaGrid Annual Conference (ChinaGrid), 2012, pp. 43 –48.

[7] Apache Hadoop, http://hadoop.apache.org.

[8] D. Borthakur et. al., "Apache hadoop goes realtime at facebook," in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2011), ACM, New York, USA, 2011, pp. 1071–1080.

[9] Hadoop Distributed File System (HDFS) http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html

[10] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM 51(1).

[11] M. Assunçãoa, R. Calheirosb, S. Bianchic, M. Nettoc and R. Buyyab, "Big data computing and clouds: trends and future directions," Journal of Parallel and Distributed Computing, vol. 79–80, pp. 3–15, May 2015.

[12] Apache Hadoop, http://hadoop.apache.org.

[13] Hadoop Distributed File System (HDFS), http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html

[14] H. Lu, H-S Chen and T-T Hu, "Research on hadoop cloud computing model and its applications," in Third International Conference on Networking and Distributed Computing (ICNDC), Hangzhou, China, 21-24 Oct. 2012.

[15] K. Krish, M. Iqbal, M. Rafique and A. Butt, "Towards energy awareness in hadoop," Fourth International Workshop on Network-Aware Data Management (NDM), 2014, pp. 16-22.

[16] S. Kulkarni, "Cooling hadoop: temperature aware schedulers in data centers," Auburn University, Alabama, 2013.

[17] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," ACM SIGOPS Operating Systems Review 2010, vol. 44, no. 1, pp. 61-65.

[18] L. Barroso and U. Holzle, "The case for energy proportional computing," IEEE computer society, 2007.

[19] H. Amur, V. Gupta, G. R.Ganger, M. A. Kozuch and K. Schwan, "Robust and flexible power proportional storage," SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing, 2010, pp. 217-228.

[20] B Lin, S Li, X Liao, Q Wu, and S Yang, "eStor energy efficient and resilient data center storage," International Conference on Cloud and Service Computing (CSC), 2011, pp. 366 – 371.

[21] B. Fan, W. Tantisiriroj, L. Xiao and G. Gibson, "DiskReduce: RAID for data-intensive scalable computing," PDSW '09 Proceedings of the 4th Annual Workshop on Petascale Data Storage, 2009, pp. 6-10.

[22] Z. Cheng Z. Luan, Y. Meng, Y. Xu, D. Qian, A. Roy, N. Zhang and G. Guan, "ERMS: an elastic replication management system for HDFS," IEEE International Conference on Cluster Computing Workshops 2012, pp. 32-40.

[23] Y. Wei, Y. Foo, F. Chen and K. Lim, "Reliable and efficient distributed storage with LDPC code," International Conference on Cloud Computing Research and Innovation, 2014.

[24] R. Kaushik, M. Bhandarkar and K. Nahrstedt, "Evaluation and analysis of GreenHDFS: a self-adaptive, energy-conserving variant of the hadoop distributed file system," IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 2010, pp. 274 - 287.

[25] R. Kaushik, T. Abdelzaher, R. Egashira and K. Nahrstedt, "Predictive data and energy management in GreenHDFS," International Green Computing Conference and Workshops (IGCC), 2011, pp. 1-9.

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

[27] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, vol. 14, pp. 76-86.

[28] J. Gao, "Machine learning applications for data center optimization," Research at Google, 2014.

[29] D. Svozil, V. Kvasnickab and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," Chemometrics and Intelligent Laboratory Systems, 1997, vol. 39, pp. 43-62.

[30] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," Computer, 1994, vol. 27, pp. 17-26.

[31] S. Ronald, "Robust encodings in genetic algorithms: a survey of encoding issues," IEEE International Conference on Evolutionary Computation, 1997, pp. 43-48.

[32] K. Stanley, and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Evolutionary Computation, 2002, pp. 99-127.

[33] T. Pencheva, K. Atanassov and A. Shannon, "Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets," Proceedings of the 10th International Workshop on Generalized Net, 2009.

[34] T. Pencheva, K. Atanassov and A. Shannon, "Modelling of a roulette wheel selection operator in genetic algorithms using generalized nets," Bioautomation, 2009, 13 (4), pp. 257-264.

[35] W. Li, H. Yang, Z. Luan and D. Qian, "Energy prediction for mapreduce workloads," IEEE Ninth International Conference on Dependable, Automatic and Secure Computing, 2011.

[36] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, S.H. Chung, and Y. Li., "Cloud computing resource scheduling and a survey of its evolutionary approaches," ACM Computing Surveys, vol. 47, no. 4, Article 63, pp. 1-33, Jul. 2015.

[37] A. Engelbrecht, Computational Intelligence: An Introduction. Willey, 2nd Edition, 2007.

[38] C. Bishop, Neural Networks for Pattern Recognition. Clarendon Press, 1st Edition, 1996.