ISSN 2278-3091
Volume 9, No.1, January – February 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse15912020.pdf
https://doi.org/10.30534/ijatcse/2020/15912020

Growing Self-Organizing Map through a Hybrid Algorithm of Load Prediction

Nawaf Alharbe[1] Fawaz Alharbi[2]
[1] **College** of Community, Taibah University, Badr, Kingdom of Saudi Arabia
anawafrasheed@gmail.com
[2] Huraymila **College** of Science and Humanities, Shaqra University, Saudi Arabia
fawazharbi@gmail.com

ABSTRACT

Big data science become hot research topic due to the huge of data are extracted every day. Big data analysis is used into many domains such as manufacturing, education, media, insurance, Internet of Things (IoT), and Healthcare. Some of these fields are required a real-time analysis. The paper proposes an enhancement of Growing Self-Organizing Map (GSOM) model. A neural network algorithm was used to cluster the input of real data. Best threshold was needed for stream processing systems are computed based on study of historical stream processing system, and chose the best strategies that met the minimum running time complexity. The achievements of the proposed algorithm experimental shows superior result compared with other prediction algorithm in both criteria time and accuracy complexity.

Key words: IoT, GSOM, Load Predicting and Stream Processing.

1. INTRODUCTION

Many systems deal with big data appear to process and handle real-time data [1-2]. Enhancing the performance of operating system in minimizing the running time as much as possible and keep stability, the operations of the system become one of the most problem that researcher trying to deal with. Load predicting is a common technique are used in predictions [2-4], which lead to be concerned by researcher as well. Traditional processing system suffering of unhandled the scalability of data processed as well the complexity of streaming data. Therefore, the load predicting of stream processing are used to overcome these problems; were needs a certain requirements for real-time system to meet the reasonable performance. Predicting model categorized as linear and nonlinear. Generally, linear prediction includes ARMA [3] and FARIMA [4] models, whereas non-linear prediction predominantly contains Neural Network NN [5], wavelet theory [6] and Support Vector Machine SVM [7]. Most flow data processing recently is categorized as unstructured data that mean the processing most probably need a non-linear prediction to manipulate these flow of data. Thus, the research spot toward non-linear methods to handle real-time data flow [8]; which mainly predict the new flow data according to rules that built previously based on analyzing of historical data [8]. Reflecting that concept, a successfully model have been made by Box-Jenkins's [9] capable to utilize the amount of data that loaded into processor, the model predicts processor behavior with the help of time need for each slice to be executed and assume it as correlated a parameter. Warrm Smith and Parkson Wong [10] proposed a prediction system based on queue structure to compute the time latency for each performed job and then considered bias factor into prediction model. Rich Wolski [11] increase the CPU performance by optimizing the effectiveness of UNIX systems according to time-based nested of job execution; the proposed algorithm achieve reasonable performance but not take in consideration the features of flowed data.

In This research, a minimize running time complexity using enhanced optimizing Self-Organized Map SOM is proposed. The proposed method discussed and illustrated in details in section II. Then, analyzed the consistent experiments and results are carried out in Section III. Finally, conclusion and discussion of this research summarized in Section IV.

2. PROPOSED METHODOLOGY

Self-organization map (SOM) considers one of the most clustering algorithms which is in pattern recognition domain [12-15]. The research aims to increase the accuracy of prediction to minimize time complexity of data streaming process. However, the simple SOM comprises three main steps; competition, cooperation and adaptation processes. In first phase, there are function compute most close neuron by calculate the Euclidean distance algorithm. equation1 represent the computation process where the n is number of output layer.

$$i(\hat{x}) = argmin_j \| \hat{x} - \hat{w} \|, j = 1,2, \ldots . n \quad \ldots\ldots\ldots\ldots (1)$$

Here n represents the quantity of output neurons. While co-acting the adjoining neurons function with harmony with each other. Weight vectors in the nearby winning neurons are adjusted. Equation (2) shows the Gaussian function. With the help of equation 3, the neighbourhood function reflects are computed. Here, σ_0 shows the initial neighbourhood radius. τ represents the time constant.

$$h_{i(x)}(n) = e^{-\frac{d_{i,j}^2}{2\sigma^2(n)}} \quad (2)$$

$$\sigma(n) = \sigma_0 e^{-\frac{n}{\tau}} \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The adaptation phase deals with the output of previous phase after the neighborhood nodes are determined. Then, the vector weight are

updating continuously using Equation (4). Here w_j represent j neuron's weight. Maximum vector value is represented by *t*.

$$w_j(n+1) = w_j(n) + \eta(n) h_{j,i(x)}(n) \left(X(n) - w_j(n)\right), n = 1,2 \quad (4)$$

$$T_\eta(n) = \eta(0)\left(1 - \frac{n}{T}\right), n = 1,2 \ldots, T \quad (5)$$

With the help of Equ 5, learning efficiency is calculated. Here η is constant number between zero and one. Number of iterations are represented by *T*. η (0) represents the initial learning efficiency. The algorithm features are extracting after three main functions are performed; sampling, similarity matching and updating respectively.

2.1 Improved Growing Threshold Setting Method

In this research, the aim is to increase the accuracy of streaming process system prediction is greater than the real time response requirements, due to the current phase output will be the input of next phase and will highly defect effectiveness of accuracy rate which finally leading to effect the response speed as well. The proposed enhancement of GSOM algorithm come in many stages. Improving the initialization network parameter, clustering prediction mode, bias node initialization and operation efficiency [16-18]. Which achieve most probably demand of loading prediction algorithm for streaming processing system [19-21].

To set reasonable value of GT, GSOM decides adding of a new neuron based on the growth threshold (GT). A too trivial threshold will cause adding neuron frequently and hence training burden will be increased. In the same way, with the too huge threshold inaccurate load will be predicted. With the growing network, neurons should be added cautiously. Hence, threshold value should be dependent on the current network condition. As per the clustering technique, values from the similar class should be as close as possible [22-25]. Besides, the values from different class can be as distant as possible. The research proposes a novel way for adjusting threshold dynamically.

$$j = arg\ arg\ \| x_n - w_j \|, j = 1,2, \ldots, m \quad \ldots\ldots\ldots\ldots (6)$$

$$GT = min \| w_i - w_j \|, i = 1,2, \ldots m, i \neq j \quad ((4))$$

Here T is the quantity of winner neuron. wj is the weight of winner neuron. The closest neighbor of neurons weight represented by *wi* where the GT resigned into the distance of neuron j and it's closest neighbori. The fact of the method is it's to guarantee where the distance from xi to wjis atleastthan the closest neighbor. The growing threshold computed by Equation(10):

$$GT < min \| x_n - w_j \|, j = 1,2, \ldots\ldots m \ldots\ldots\ldots\ldots\ldots (7)$$

The competition neurons counted by *m*. where x node considered as input to the network and then grows itself once the distance between the x and it's winner neuron *j* larger than the threshold of growing rate.

2.2 Initial Parameter Optimization

2.2.1 Neuron Number Initialization Algorithm

The network dynamically adds neurons and adjusts parameters till it gains steady states with each new input pattern, [26-28]. Hence, it will frequently add neurons during training phase if initially there is too small quantity of neurons. This, in turn, will influence the system's response time. Similarly, a large number of neurons will lead to death of neurons. This in turn, will negatively affect the training process [29-31]. Hence, it is expected that a sufficient quantity of initial neurons will help SOM network to gain momentum. The algorithm-1 is based on the idea of dichotomy. In the algorithm, *distmean*, the average distance, is calculated for the input sample set X. Here, a Euclidean distance distij smaller than average distance for the Xi , Xj shows the likelihood of both the inputs to belonging to the same class [32-35]. By A class of input vectors is pre-processed by using probabilistic analysis and dichotomy method. Here, M is found to be used as the number of initial neurons. In contrast to the traditional methods which form its basis on experience or choosing a fixed m, the proposed technique is expected to speed up the training process and hence minimize the number of iterations. The algorithm is described as algorithm 1.

Algorithm 1 Neuron number initialization algorithm

Input:
Set of Input X

Output:
Number of initial neurons m

1: Initialization compute the mean distance of all X_i in X; set up an empty array $Cate$ to record the patterns
2: Choose a X_i from X;
3: For each X_j in $Cate$, compute distance between X_i and X_j in , if $dist_{ij} < dist_{mean}$, then X_i is considered to belongs to the category represented by $Cate[j]$;
4: If X_i doesn't belongs to any knowing categories, add it to $Cate$;
5: Repeat step 2 to step 4 until X is empty;
6: Set $m = sizeof(Cate)$;
7: **return** m;

2.2.2 Initialization of Weights of Neurons

The first step is to initialize the neurons weights. These neurons are adjusted for reflecting features of input data class during the training process. Traditionally, in SOM networks neuron's weights have been initialized with random numbers. This is the reason, these weights do not exhibit any feature of the training data. The paper has proposed a method to initialize neurons with typical input set. Since the quantity of neurons is m, initializing the weights of m neurons becomes an issue while getting m input set for representing the features of their respective class. In general the objective of clustering is to bring closer the nodes in the same class, and to make as far as possible the nodes of different class. In the proposed work, greedy algorithm has been used for selecting m vectors from the input data set which are most distant from each other. Finally, the neuron's weights are initialized with these sets.

1.2.3 Optimization of Computational Performance

During the training process, repeated iterations are performed in SOM. Further, with the addition of new neutron each time, weights of the complete network are adjusted. The prediction being timeliness, complexity of load prediction through traditional SOM algorithm is intolerable. A caching-based load prediction technique has been introduced by the researcher to solve this problem. SOM has been used as a classifier for accurately predicting the load in the proposed prediction technique. In addition, computational efficiency of load prediction has also been improved with the help of following three methods:

1.2.3.1 **Strategy for Assignment of Neuron Weight Vector**

The initial value of network connection weight largely influence the efficiency of the learning process of the network. To accelerate retraining process after addition of a neuron, the weight of winning neuron as well as the input pattern itself need to be used for assigning new neuron node's weight. Formula for weight initialization has been given as:

$$w_{new} = a * w_i + b * X_i + c * Random \quad \ldots\ldots\ldots\ldots (8)$$

Here w_{new} represents linear combination of the winning neuron weight w_i and the new pattern vector X_i. For ensuring that the weight does not bias the current vector X_i, a random number is assigned. As per the experimentation results, its performance is good when a is 1/5, b is 3/5 and c is 1/5. New neuron's initial weight need not be exact, as it is continuously updated in further iterations. Moreover, a reasonable beginning value will aid in reducing the iterations. Further, it will stabilize network.

1.2.3.2 **Strategy of Prediction**

An input vector which does not conforms to the constrains of winning neuron reflects that the distance of input vector and wining neuron is greater than the threshold. In this case, a new neuron is added to the network. But no known load information is there in the new empty neuron. For solving the problem, the author has proposed a prediction mechanism. An input belonging to a known cluster, predicts the load of that cluster based on the historical data of the same. In other cases, the load is predicted with the help of linear regression algorithm depending on all the historical data. When a real load arrives, the new neuron's information is updated. Working of the linear regression algorithm is as follows:

Suppose there is an input matrix X. The vector w contains the regression coefficient. Prediction outcome is given as $Y = X^T w$. Use of square error to measure the effect will lead to best prediction:

$$Err = \sum_{i=1}^{m} (y_i - x_i^T \omega)^2 \ldots\ldots 9$$

The same can also be shown in terms of a matrix as $(y - Xw)T(y - Xw)$. After obtaining the w's derivatives its value is put as zero. Then w is obtained as follows:

$$\hat{\omega} = (X^T X)^{-1} X^T y \ldots\ldots\ldots\ldots\ldots 10$$

Where weight vector is ˆ w and the input data set is X.

2.4 Cold Backup Strategy

As discussed, addition of new neurons causes the SOM network to reiterate for adjusting parameters. In addition, as the iteration process has higher complexity, it will not meet the real-time requirement of stream processing system. To solve the issue, the author has proposed the SOM cold backup strategy. For the same, dynamic as well as static SOM netwokrs are created. The static network receives input and predicts the load whereas the dynamic network adds new neurons and retrains it for making the network stable. The two networks work in synergy as follows:

Algorithm 2 Load Predicting algorithm based on Improved Growing SOM

Input:
Input vectors set X consisting of data and compute topologies.
Output:
Prediction of load for a specific X_i

```
Initialization Compute initial neuron number m and network weights W according to the training data set.
Training the network to adjust the weights.
for each X_i ∈ X do
    Compute its wining neuron w_j;
    Compute current network's growing threshold GT;
    if distance between X_i and w_j less than GT then
        Take historical data from the cluster X_i belongs to and compute the load;
    else
        Add new neurons dynamically, retraining the dynamic network;
        Predict load with linear regression algorithm and return the value;
    end if
    Get real load of X_i after task_i is done.
    Update the training data set.
end for
```

- Initially, both networks are the same.
- On the arrival of input vector X*i*, the winning neuron is computed by SOM and is compared with the threshold GT. Historical data is taken to predict load for input X_i if in case the input is from an existing cluster.
- The dynamic SOM network sums up neurons and holds the network parameters if X*i* is from a new cluster. Now the static network be same, and calculates the results with the help of linear regression algorithm.
- As soon as the dynamic SOM completes retraining process, dynamic network is replicated for replacing the static classifier.
- Training iteration leads to the dynamic SOM network. By this way, the problem of failure to satisfy the real-time requirement because of network updates.

3 IMPLEMENTING THE IMPROVED LOAD PREDICTING ALGORITHM

The available GSOM algorithm helps to dynamically add neurons. In addition, it also helps to identify new classification of input vectors [14]. But frequent iterations of the algorithm and computing speed does not allow to satisfy the requirement of the real-time performance of the stream processing system. To overcome the issue the author presents a Load Predicting based on Improved Growing Self-Organizing Map (LP-IGSOM) algorithm for recognizing input task' cluster and to foresee its' load requirement precisely and timely. The LP-IGSOM has improvement over existing GSOM in terms of initialization of neuron numbers, optimization of calculated efficiency etc. The LP-IGSOM algorithm has been shown as Algorithm 2:

The algorithm has three main phases:
1) Initialization phase:
a) As per the known input mode, rough class number m is calculated for initializing the neuron's quantity.
b) m most distant input vectors are chosen and weight of m neuron are initialized.
c) The growing threshold is calculated as per the current network status.
2) Growth phase:
a) Input is added to the network.
b) Euclidean distance is used for finding the winning neuron for traditional SOM algorithm.
c) if winning neuron > threshold GT. if not, step f is followed.
d) A neuron is added in case winning neuron is a boundary node and its weights are initialized using the current input mode X, the weight W of the winning neuron, and the random value. If not, step f is followed.
e) Learning rate is again set to the initial value and then adjusted to the neighborhood to the initial value.
f) Neighborhood vector of neurons is updated.
g) Step b to step f are repeated until the clustering effect is stabilized for the existing data.
3) Prediction phase:
a) Winning neuron is obtained. If no need is there to add new nodes then all the known loads from winning neuron is taken and average load prediction is computed.
b) If addition of a new node is required, linear regression is used by the static network to predict the load on the input mode. In addition, addition of new nodes and retraining is done by dynamic SOM network.
c) Real information of the load is visited to add to the new neurons.
The training of SOM network is based on existing data sources, computing topology and load data sets. When a new computational task is coming, calculate whether the Euclidean distance in the SOM network satisfies the

constraint, and if so, perform SOM clustering to match similar loads and make predictions. Due to the characteristics of data streams, different computing topologies often arise from new data sources. Therefore, when the Euclidean distance calculated by the above algorithm does not meet the constraints, the linear prediction method is used as a supplement. In this case, according to the load of computing nodes, the competitive neurons in the neural network can be updated to improve the accuracy and prediction range.

3.1 Data Analysis and Results

The data used in the paper is the simulation data obtained by the pavement sensor network. This data coming at the speed of 5000 pieces per second.

Table 1: Message classification

Message classification	Message contents
Sensor health information	Sensor number, the road, Energy status, health status
Daily monitoring information	Sensor number, License plate number, the current speed
Vehicle tracking information	Sensor number, target license plate number, vehicle characteristics, running track, driver characteristics

As shown in Table 1, the sensor data stream mainly contains three types of information, each subclass contains an optional fixed message content. Simulate data sources simulate the sensor's message queue by randomly generating vehicle data. Analog data sources can generate a wide variety of user-defined types of data. In a typical stream processing system, the user generates computational topologies by dragging operator combinations. Besides the data sources, the factors that influence the computation load include the topology generated by the submission task, which defines how the data message is processed and the corresponding input mode. To fully test the dynamic neuron-joining capabilities of this chapter based on the improved GSOM load predicting algorithm, source data and task topology were customized. Divided into two categories:

(1) Standard data source and fixed computing topology.

(2) Standard data sources and customized computing topology.

In the second category, on the basis of fixed computational topology, a variety of different computing rules are continuously simulated to generate different computing topologies, so as to test the addition of dynamic neurons.

3.2 Experimental results

Since the data stream by nature shows particularity, a little work is found on load predicting. The classic linear regression prediction algorithm and the classical clustering algorithm K-means has been chosen to compare with the proposed work. The load on the data stream sent by the sensor network is predicted. The same is then compared with the real load. The parameters for experimentation has been given as:

1. The parameter for maxlearning rate is set as 0.9. The parameter for min learning rate is set as 1E-5.
2. 1000 is the total iterations for training the network.
3. 5 has been set as the initial neighbourhood radius.

Figure 1 depicts the load change in actual situation with time while stream processing system operates.

Samples has been taken for every two minutes in the range 0-20 with standard data source and fixed computing topology. Calculated load by GLP-SOM is predicted and linear regression, k-means clustering is listed. Comparison has been made with the real load and shown in Figure 2.

The input modes are known with fixed computing topology. Hence, linear regression, K-means along with GLP-SOM can more accurately predict the load which can be seen Figure 2. Actual load curve and predicted curve is almost similar. Her, 81 is the MSE of LR algorithm is 81. 32.7 and 21.5 are the the errors of k-means and GLP-SOM respectively. When data source fluctuates, LR algorithm exhibits a relatively poor prediction effect. Moreover, the prediction effect is comparatively stable based on clustering algorithm.

As per computation rules, corresponding to each computation modes, new computation topologies are continuously generated. Figure 3 shows that with new computation rules, the data source as well as the computation topology does not require any previous of data. The situation is not predicted well by using linear regression prediction. In this case, we got larger prediction error to the extent of 322.5. K-means of fixed clustering performs worst, and the MSE reaches 383.9 among all the three prediction classifiers. By fixing K's value, the new input mode is strongly classified into the clusters existing. It will affect the current clustering characteristics. Use of k-means will lead to the greater error in prediction effect. The proposed algorithm, in the arrival of new input, is able to recognise as well as to dynamically grow neurons. The MSE is 77.6. Hence, the overall prediction effect is comparatively accurate. As per the results of the experiment, the algorithm for load forecasting handles effectively the new input mode. The proposed algorithm outperforms the traditional methods in terms of accuracy and speed of load prediction. In some nodes, because the algorithm is still in training, the prediction accuracy lags behind and the worst effect declines to the effect of linear regression prediction, but the overall prediction effect is more satisfactory.

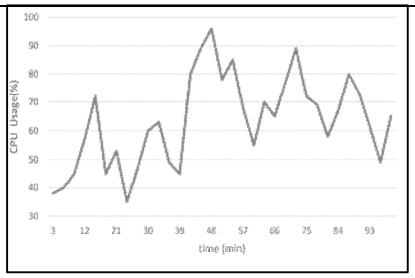

Figure 1: Nodes with Real Load

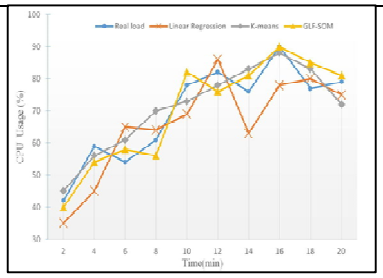

Figure 2: Load prediction with standard data source and fixed computing topology

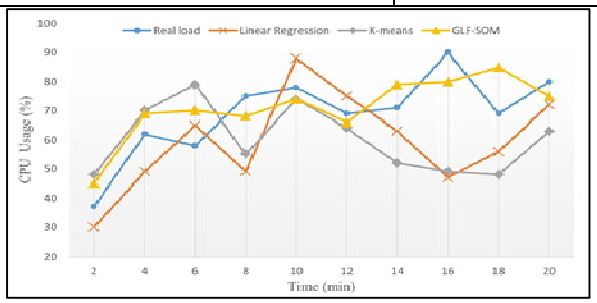

Figure 3: Load prediction with customized computing topology

4 CONCLUSION

The increasing volume of data is becoming challenging in many areas especially in continuous data stream. Handling streaming data and making real-time decisions based on the data require heavy and complex processes which go beyond traditional processing system. Thus, an algorithm for prediction of load prediction has been developed. The developed algorithm is able to handle the complex structures of the stream processing system better than other traditional algorithms for prediction. The results showed that accuracy rate of prediction and score of speed efficiency of the developed load prediction algorithm LP-IGSOM is greater than traditional load predicting algorithms. Future work may include applying the algorithm on larger volume of data and different data sources.

REFERENCES

1. Ali Salehi. Design and implementation of an e cient data stream processing system. 2010.
2. I. Moghram and S. Rahman. Analysis and evaluation of ve short-term load forecasting techniques. IEEE Transactions on Power Systems, 9(11):42{43, 1989. https://doi.org/10.1109/MPER.1989.4310383
3. TrondRiise and Dag Tjozstheim. Theory and practice of multivariate arma forecasting. Journal of Forecasting, 3(3):309{317, 2010. https://doi.org/10.1002/for.3980030308
4. YantaiShu, Zhigang Jin, Lianfang Zhang, and Lei Wang. Tra c prediction using farima models. In IEEE International Conference on Communica-tions, pages 891{895 vol.2, 1999.
5. S. Haykin and N. Network. A comprehensive foundation. Neural Networks, 2004.
6. HongYan Wen. Research on Deformation Analysis Model Based Upon Wavelet Transform Theory. PhD thesis, Wuhan University, 2004.
7. NelloCristianini, JohnShawe-Taylor, Cristianini, and Shawe-Taylor. An In-troduction to Support Vector Machines and Other Kernel-based Learning Methods. PUBLISHING HOUSE OF ELCETRONICS INDUSTRY, 2004.
8. M. T Hagan and Suzanne M Behr. The time series approach to short term load forecasting. Power Systems IEEE Transactions on, 2(3):785{ 791, 1987.
9. B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok. A resource monitoring system for network-aware applications. In Proceedings of the 7th IEEE International Symposium on High Perfor-mance Distributed Computing (HPDC, 1998.
10. Warren Smith, Parkson Wong, and Bryan A. Biegel. Resource selection using execution and queue wait time predictions. Nasa Ames Research CenterTrNas, 2001.
11. R Wolski, N Spring, and J Hayes. Predicting the cpu availability of time-shared unix systems on the computational grid. Cluster Computing, 3(4):293{301, 2000. https://doi.org/10.1023/A:1019052825453
12. J Vesanto and E Alhoniemi. Clustering of the self-organizing map. IEEE Transactions on Neural Networks, 11(3):586, 2000. https://doi.org/10.1109/72.846731
13. D. Alahakoon, S. K. Halgamuge, and B. Srinivasan. Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Transactions on Neural Networks, 11(3):601{614, May 2000.

https://doi.org/10.1109/72.846732
14. Alharbe, N. R. (2018) ‘ Load Predicting Algorithm based on improved Growing Self-Organized Map’, in The 9 The 13th International Conference on Future Information Technology (FutureTech 2018). April 23-25, 2018, Salerno, Italy.
15. Alka Agrawal, MamdouhAlenezi, Rajeev Kumar, Raees Ahmad Khan, (2019), Measuring the Sustainable-Security of Web Applications through a Fuzzy-Based Integrated Approach of AHP and TOPSIS, IEEE Access, Volume 7, 2019, pp. 153936-153951, Nov-2019.
16. Rajeev Kumar, Mohammad Zarour, MamdouhAlenezi, Alka Agrawal, Raees Ahmad Khan, (2019), Measuring Security-Durability through Fuzzy Based Decision Making Process, International Journal of Computational Intelligence Systems, Volume 12, June, 2019.
17. Alka Agrawal, MamdouhAlenezi, Rajeev Kumar, Raees Ahmad Khan, (2019), Source Code Perspective Framework to Produce Secure Web Application, Computer Fraud and Security, Elsevier, Available at Thomson Reuters, October 2019.
18. Kavita Sahu, Rajshree, Rajeev Kumar, Risk Management Perspective in SDLC, International Journal of Advanced Research in Computer Science and Software Engineering, pp. 1247-1251, 2014.
19. Alka Agrawal, Mohammad Zarour, MamdouhAlenezi, Rajeev Kumar, Raees Ahmad Khan, (2019), Security durability assessment through Fuzzy Analytic Hierarchy process, PeerJ Computer Science, 2019. https://doi.org/10.7717/peerj-cs.215
20. Rajeev Kumar, Suhel Ahmad Khan, Alka Agrawal, Raees Ahmad Khan, (2018), Security Assessment through Fuzzy Delphi Analytic Hierarchy Process, ICIC Express Letters-An International Journal of Research and Surveys, Volume 12, Number 10, pp. 1063-1069.
21. Alka Agrawal, MamdouhAlenezi, Suhel Ahmad Khan, Rajeev Kumar, Raees Ahmad Khan, (2019), Multi-level Fuzzy System for Usable-Security Assessment, Journal of King Saud University - Computer and Information Sciences, April 2019 https://doi.org/10.1016/j.jksuci.2019.04.007.
22. Rajeev Kumar, Suhel Ahmad Khan, Alka Agrawal, Raees Ahmad Khan, (2018), Measuring the Security Attributes through Fuzzy Analytic Hierarchy Process: Durability Perspective, ICIC Express Letters-An International Journal of Research and Surveys, Volume 12, Number 6, pp. 615-620.
23. Alka Agrawal, MamdouhAlenezi, Dhirendra Pandey, Rajeev Kumar, Raees Ahmad Khan, (2019), Usable-Security Assessment through a Decision Making Procedure, , ICIC Express Letters-Part B, Applications, IEEE, Volume 12, Number 9, 2019.
24. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2016) Durability Challenges in Software Engineering, Crosstalk-The Journal of Defense Software Engineering, pp. 29-31.
25. MamdouhAlenezi, Rajeev Kumar, Alka Agrawal, Raees Ahmad Khan, (2019), Usable-Security Attribute Evaluation using Fuzzy Analytic Hierarchy Process, ICIC Express Letters-An International Journal of Research and Surveys, Volume 13, Number 6, 2019.
26. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2015) Revisiting Software Security: Durability Perspective, International Journal of Hybrid Information Technology, Vol.8, No.2 pp.311-322. (Science & Engineering Research Support Society) (Indexed in SCOPUS). https://doi.org/10.14257/ijhit.2015.8.2.29
27. Kavita Sahu, R. K. Srivastava, Revisiting Software Reliability, Data Management, Analytics and Innovation, Advances in Intelligent Systems and Computing, Springer, pp. 221-235, 2019.
28. Kavita Sahu, R. K. Srivastava, Soft Computing Approach for Prediction of Software Reliability, ICIC Express Letters, pp. 1213-1222, 2018.
29. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2016) Analytical Network Process for Software Security: A Design Perspective, CSI Transactions on ICT, Springer, Volume 4, Issue 2–4, pp. 255–258, 2016. https://doi.org/10.1007/s40012-016-0123-y
30. Kavita Sahu, Rajshree, Helpful and Defending Actions in Software Risk Management: A Security Viewpoint, Integrated Journal of British, pp. 1-7, 2015.
31. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2014) Software Security Testing: A Pertinent Framework, Journal of Global Research in Computer Science, Volume 5, Issue 3, pp. 23-27, 2014.
32. Kavita Sahu, Rajshree, Stability: Abstract Roadmap of Security, American International Journal of Research in Science, Engineering & Mathematics, pp. 183-186, 2015.
33. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2015) Revisiting Software Security Risks, British Journal of Mathematics & Computer Science, Volume 11, Issue 6, 2015. https://doi.org/10.9734/BJMCS/2015/19872
34. Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, (2014) Software Security Durability, International Journal of Computer Science and Technology, Vol. 5, Issue 2, pp. 23-26.
35. Kavita Sahu, Rajshree, Software Security: A Risk Taxonomy, International Journal of Computer Science & Engineering Technology. pp. 36-41, 2015.