UDC 519-62
DOI: 10.20535/SRIT.2308-8893.2022.3.06

RESOURCE SCHEDULING IN EDGE COMPUTING IOT NETWORKS USING HYBRID DEEP LEARNING ALGORITHM

G. VIJAYASEKARAN, M. DURAIPANDIAN

Abstract. The proliferation of the Internet of Things (IoT) and wireless sensor networks enhances data communication. The demand for data communication rapidly increases, which calls the emerging edge computing paradigm. Edge computing plays a major role in IoT networks and provides computing resources close to the users. Moving the services from the cloud to users increases the communication, storage, and network features of the users. However, massive IoT networks require a large spectrum of resources for their computations. In order to attain this, resource scheduling algorithms are employed in edge computing. Statistical and machine learning-based resource scheduling algorithms have evolved in the past decade, but the performance can be improved if resource requirements are analyzed further. A deep learning-based resource scheduling in edge computing IoT networks is presented in this research work using deep bidirectional recurrent neural network (BRNN) and convolutional neural network algorithms. Before scheduling, the IoT users are categorized into clusters using a spectral clustering algorithm. The proposed model simulation analysis verifies the performance in terms of delay, response time, execution time, and resource utilization. Existing resource scheduling algorithms like a genetic algorithm (GA), Improved Particle Swarm Optimization (IPSO), and LSTM-based models are compared with the proposed model to validate the superior performances.

Keywords: edge computing, cloud computing, Internet of Things (IoT), resource scheduling, deep learning.

INTRODUCTION

The Internet and smart devices have become indispensable elements in daily life. People depend on their smart devices for daily activities like payment, healthcare, virtual reality, games, etc. These different applications increase the resource requirements of smart devices. Cloud computing has been adopted to meet resource demands. The cost-effective cloud solutions offer numerous advantages in information technology and ensure that users receive essential computing, storage, and communication services based on their needs [1]. The tremendous applications based on IoT networks enhance the quality of life (Fig. 1). However, the high bandwidth requirement for IoT applications increases energy consumption, transmission bandwidth, and delay. Moreover, providing all the user-requested services using cloud computing is difficult since more than ten billion edge devices are deployed every day, and the rate is increasing [2]. To overcome these issues, edge computing paradigms have been introduced.

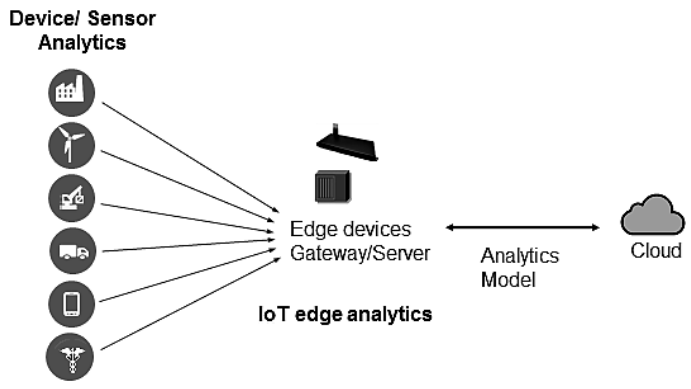

Fig. 1. IoT edge networks

Edge computing IoT networks allow the process to be performed near the device or node. The user-requested functions and services from the edge cloud are moved near to the user in edge computing to provide better storage and computing facilities [3]. Figure 1 depicts a simple illustration of IoT edge networks.

Though edge computing provides a wide range of services in various domains, it is essential to look into resource management. The increased number of user services increases the bandwidth demand on IoT networks. The resource scarcity problem and the computational complexities of IoT networks reduce the overall quality of services. Edge computing provides user-requested resources incorporating multiple techniques like clustering algorithms, and scheduling algorithms to define the user demand. Since the resource requirements to compute data types in IoT systems are different, the data processing is generally performed at regular intervals. In order to process diverse data, the system requires different computation resources, which should be provided by edge computing by switching resources from one to the other. This process will increase the computation time. To avoid this, clustering techniques are used in edge computing before allocating the resources [4].

Traditional clustering techniques like *k*-means, fuzzy c-means, hierarchical clustering, etc. are used in various research models. However, the conventional methods lag in performance while handling large data volumes. Moreover, conventional clustering techniques require additional dimensionality reduction techniques, which increase the overall computation cost [5]. Considering this limitation, in our previous work, we have employed an improved spectral clustering algorithm that clusters the resource requirements based on the data similarity [25]. The process flow of the clustering model is simply illustrated in Fig. 2. Compared to conventional cloud computing services, edge computing will provide minimum latency and support a wide range of IoT applications. As discussed, to satisfy the computation requirements of IoT and improve the quality of services, a hybrid deep learning-based resource allocation procedure is presented in this research work. Summarized research contributions are presented as follows.

- Presented a hybrid deep learning model for resource allocation in edge computing using deep bidirectional recurrent neural network and convolutional neural network technique.

- Presented an intense experimental analysis of the proposed model in terms of different metrics like resource utilization, response time, execution time, delay, and efficiency.

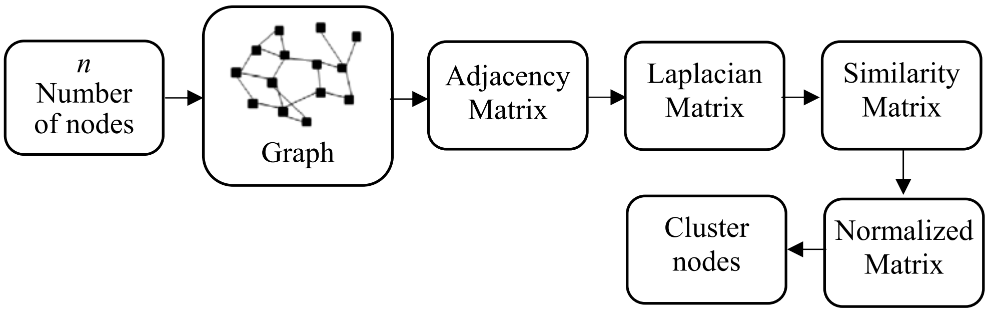

Fig. 2. Improved spectral clustering algorithm

- Presented a comparative analysis of the proposed model to validate the superior performance with conventional techniques like Improved PSO (IPSO), Genetic Algorithm (GA), and LSTM based resource scheduling procedures.

The remaining part of the article is arranged in the following order: Literature analysis of existing scheduling approaches are presented in section 2. Section 3 presents the hybrid deep learning based resource scheduling model. Section 4 presents the details of simulation results and discussion and finally, the features are concluded in section 5.

RELATED WORKS

Resource scheduling strategies that have evolved in the past few years are considered for literature analysis, and the observations are summarized in this section based on the methodology, feature merits, and demerits. The recent trends in resource scheduling in edge computing are analyzed in [6] based on resource allocation, computation offloading, and resource provisioning. The techniques that have evolved for scheduling are categorized into centralized and distributed approaches. Applications related to these centralized and distributed approaches are analyzed and discussed in detail, which provides a basic ideology about the resource scheduling procedures. A hybrid resource scheduling procedure in edge computing was reported in [7] as a four-layer computing system that supports intelligent operations in a smart manufacturing environment. The presented two-phase hybrid algorithm incorporates greedy and threshold strategies for resource scheduling to minimize energy consumption and maximize efficiency in a manufacturing environment.

A dynamic scheduling approach for edge computing was reported in [8] incorporates deep reinforcement learning and deterministic policy gradient methods to minimize delay, energy consumption, and cache fetching costs. The presented learning models schedule the resources based on cache, offload status for uncached tasks, offloading transmission power, and edge computing resource status. The combined approach minimizes the cost function and performs better than conventional deep Q networks. The major objective of edge computing is to provide suitable computing resources for user requests in a static and dynamic environment. The issues in resource allocation are formulated as a nonlinear optimization problem in [9] and presented with a regularization-based agnostic online algorithm. The presented approach split the major issues into subcategories and formulated an objective function for each subcategory. Convex programming is used to solve the objectives, and the experimental results validate the better performance over the online greedy one-shot solution.

A stochastic optimization was formulated in [10] considering the resource scheduling and offloading in local devices, back-end cloud and base station. To attain the objective of minimum energy consumption, and meet the QoS requirements, the stochastic optimization problem is converted into a dynamic optimization problem. To obtain an optimal solution to the dynamic optimization problem, a Lyapunov optimization theory based offloading and scheduling procedure was presented, which improves the overall performance and minimizes energy consumption. Similarly, a delay based Lyapunov function was utilized in [11], [12] to minimize the scheduling delay in multi-server edge computing systems. Without traffic statistics, the computation and communication delay can be formulated using the function and directly minimize the latency compared to traditional approaches.

Task offloading and resource scheduling in hybrid edge cloud computing reported in [13] considers the applications as graphs and analyzes the resource requirements to minimize the rent cost, time, and energy. Semidefinite and dual decomposition methods are employed to obtain offloading and scheduling decisions. Furthermore, the deep reinforcement learning model is employed to obtain discrete offloading decisions and computation frequencies. Better convergence and scheduling performance are the observed features of the presented work. The task offloading and resource scheduling procedure reported in [14] employs a decomposition model to reduce the computation complexity of the system. Logic-based bender decomposition is employed to obtain the optimal solution for master and subcategorized issues in edge computing resource management. The presented model attains better performance in delay-sensitive applications compared to conventional approaches.

An optimal task offloading and scheduling process reported in [15] considers the completion latency and energy consumption to frame the research objective. Based on the Markov decision procedure, a reinforcement learning model is employed in the presented work. Initially, it converts the problems of dynamic network conditions and task generations into a decision process. Then a double Q network is utilized along with a neural network to define the rewards attained by the system. The presented approach additionally includes a context-aware attention mechanism that assigns different weights to each action to validate better performances. Mobile edge computing, proximity-aware task offloading and scheduling were presented in [16]. The research model initially considers the distributed resources, user mobility, energy requirements, and task properties as a mixed integer non-linear programming problem. The optimal solution for the given problem is obtained using a genetic algorithm and a heuristic mobility-aware scheduling scheme was presented for effective task assignment with minimum delay and energy constraints compared to traditional methodologies. A deadline-aware task dispatching and scheduling model presented in [17] is used to schedule the new tasks and take the decision to replace the existing tasks to meet the deadline constraints. The non-trivial analysis provides better scalability compared to centralized algorithms. The presented approach minimizes the latency in sensitive applications and reduces the deadline miss ratio compared to traditional approaches.

Collaborative offloading and resource allocation algorithms reported in [18] improve the overall performance of edge computing systems and make sure that the response time limits are met. Presented migrating birds optimization algorithm identifies the optimal solution for the resource allocation problem considering the memory, CPU, energy, task queue, and servers. Maximizing the service rate with better load balancing and minimizing energy requirements are the observed features of the presented research model. A similar collaborative task scheduling model for edge computing IoT networks was reported in [19], which defines the offload state based on energy consumption and execution time. The presented approach defines when to execute the offload tasks based on the local task execution, which improves the overall throughput and deadline satisfaction ratio for critical tasks.

The resource allocation procedure for vehicle-mounted edge computing reported in [20] employed a piecewise linear approximation and relaxation procedure to obtain the optimal solution. Further, a gap-adjusted branch and bound algorithm are presented that includes a lookahead branch scheme to improve the scheduling performance over conventional scheduling models. A similar vehicular edge computing model reported in [21] incorporated the Markov decision process into the deep reinforcement learning model to obtain better training efficiency. The presented deep reinforcement learning model implementation is defined based on the proximal policy optimization algorithm. A convolutional neural network was also incorporated to approximate the value and policy functions to extract the essential features. From the literature analysis, it can be observed that the features of deep learning algorithms are not effectively utilized in resource scheduling. Most of the scheduling procedures still follow the linear and non-linear optimization problem, and solution practices. Moreover, the performance of such optimization models is also not up to the mark. Reinforcement learning is widely used in resource scheduling, but the architecture can lead to an overload state and diminish the results. Considering these limitations, a hybrid deep learning-based resource scheduling model is presented in this research work in the following section.

PROPOSED WORK

The proposed resource scheduling model is developed using a bidirectional recurrent neural network model which includes a 1D-convolutional neural network model, recurrent network, and fully connected neural network block. Instead of a conventional recurrent neural network, the presented model is combined with a convolutional neural network to obtain better performances. The included convolution block is trained to learn the features from input data. The resource requirements are considered as input data and details of resources are formulated as complete information using a one-hot encoding procedure. Then the time series resource requests are processed using a bidirectional recurrent neural network model which includes a long short-term memory unit. Finally, the fully connected neural network is used to sample the output which improves the scheduling performances.

Initially, the convolutional neural network receives the requested resource details as input which includes the features like resource category, duration, subclass, etc., Improved spectral clustering algorithm which is utilized in our previous work is incorporated in this model also to cluster the resource requirements. These clustered information features are converted into quantified information using the encoding procedure. Instead of an integer encoding procedure, one-hot encoding is used in the proposed work. Since in integer encoding if the requirements are encoded in natural order there may be a chance for featuresimposed which affect the performances. We consider the time duration between resource requests as a distance factor in the encoding procedure and in the proposed work distance between each request is considered as same. i.e., the time duration between one request to another request is considered as same to reduce the experimental computation complexity.

One-hot encoding present the information as a one-dimensional array function with equal length where the value 1 in the array represents the specific resource request. Though the dimension of one hot encoding array is high due to the zero elements in the array which is used to fill the complete data. To manage this dimensionality issue, 1D-Convolutional neural networks are incorporated in the proposed architecture. CNN reduces the computational complexity by extracting the features using feature weights and activation functions. In the presented CNN model, rectified linear unit (ReLU) activation function is used and the procedure is formulated using filters $w = \{w_1, w_2, \ldots, w_\beta\}$ and features $f = \{f_{(1)}, f_{(2)}, \ldots, f_{(n)}\}$ as follows.

$$c_m = f_{(n+2p-m)} \times w_1 + \ldots + f_{(n+2p)} \times w_\beta ,$$

where $m = \dfrac{n + 2p - \beta}{s} + 1$;

$$f(x) = x^+ = \max(0, x) ,$$

where c_m is the convoluted value and the input size is reduced into m from n in one layer. if more number of layers are included then the size can be reduced further. Though employing CNN in the scheduling process reduces the computation cost significantly it reduces the complexities of recurrent block which includes LSTM and fully connected neural network models.

Following the 1D-CNN model, a bidirectional recurrent neural network is employed in the proposed architecture. The presented deep RNN model is the major element in the scheduling process that analyzes the resource requests effectively. Since the conventional deep neural network doesn't have the ability to preserve the learned information, it is not able to schedule the resource in the future. The process must be repeated again in order to satisfy the resource requests. These limitations are overcome by the presented RNN model, which learns the information in both directions, i.e., forward and backward, so that the system needs not be trained again for new requests. The RNN architecture is able to address the long-term dependencies. However, if the request gap is too large for the resources, then the RNN will exhibit poor performance. So, a type of RNN that performs better than conventional RNNs is employed in the proposed work.

Long short-term memory (LSTM) is a type of RNN model that is utilized as a bidirectional model to predict the user resource requests in the scheduling process. To attain better performance, the cells in the model obtain the input from the previous layers and time step as a two directional propagation. The proposed network model is bidirectional, so two activation functions are employed, which are represented as $\vec{\alpha}^t$ and $\bar{\alpha}^t$. The only difference between the activation functions is the input and output direction. Several gates like forget, update and output gates are used in the LSTM cells to controls the information flow over time steps. Two functions like sigmoid and tanh functions are used in the system design which is mathematically formulated as

$$\sigma(x) = \frac{1}{1+e^{-x}},$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}.$$

The inputs for the LSTM cell are obtained from the previous layers along with time step, activation and direction steps. The cell calculates the output through the activation function along with gates. Initially the forget gate hold or drop an information. If a resource request is set then LSTM is used to track the resource schedule over time slot. If the identified resource is need to be changed due to unavailability, then its associated previous stored values will also be removed as a update process. The forget gate is the main authority which decides which information has to be stored or removed from the memory. Consider the previous activation function as $\vec{\varepsilon}^{(t-1)}$ and its current time step is $\varphi(t)$, then the forget gate is represented as a function

$$\mathcal{C}_f^t = \sigma(w_f[\alpha^{t-1}, \varphi(t)] + \vartheta_f),$$

where the weights are represented as w_f, previous activation function is represented as α^{t-1} $\varphi(t)$, the current time step input is represented as $\varphi(t)$ and forget bias term is represented as ϑ_f. The above function results into a vector function in the range [0,1]. Further this forget gate values are multiplied element wise with the previous cell $\mathcal{C}_f^{t-1}$. If the value of the multiplication process is obtained as zero or nearer to zero then the selected resources are removed from the schedule. If the value is near to or absolutely one then the resource will be held for further process. The tanh function will create new candidate values if an information is removed by the forget gate. New candidate selection is based on the following formulation

$$\mathcal{C}_f^t = \tan h(w_{\mathcal{C}}[\alpha^{t-1}, \varphi(t)] + \vartheta_{\mathcal{C}}),$$

where $w_{\mathcal{C}}$ represents the weight function and $\vartheta_{\mathcal{C}}$ represents the bias term for tanh function. The sigmoid function in the network updates the information about new candidate. The update gate has full control over the candidate values and it will add the values to the cell state. The major process of update gate is to relate the candidate to the respective previous files. The update operation is mathematically expressed as

$$\mathcal{C}_u^t = \sigma(w_u[\alpha^{t-1}, \varphi(t)] + \vartheta_u),$$

where $\mathcal{C}_u^t$ is the vector and the values are present between [0,1]. Similar to previous operation an element wise multiplication is performed with $\hat{\mathcal{C}}^t$ to compute $\mathcal{C}^t$. w_u represents the update gate weight and ϑ_u represents the update term bias function. The information flow is decided by the forget gate and update gate decides which candidate values are need to be updated. based on this process, the new cell state is updated as follows.

$$\mathcal{C}^t = \mathcal{C}_f^t \times \mathcal{C}_f^{t-1} + \mathcal{C}_u^t \times \hat{\mathcal{C}}^t .$$

Finally, the outputs are obtained by the output gate which considers the cell state as follows:

$$\mathcal{C}_{\mathcal{O}}^t = \sigma(w_{\mathcal{O}}[\alpha^{t-1}, \varphi(t)] + \vartheta_{\mathcal{O}}]);$$

$$\alpha^t = \mathcal{C}_{\mathcal{O}}^t \times \tan h\ (\mathcal{C}^t),$$

where the output weight is represented as $w_{\mathcal{O}}$ and bias term for output is represented as $\vartheta_{\mathcal{O}}$. In the proposed method, the output can be the requested resource for schedule and subsequent resource requests which are need to be scheduled. Fig. 3 depicts the overall architecture of proposed resource scheduling model.

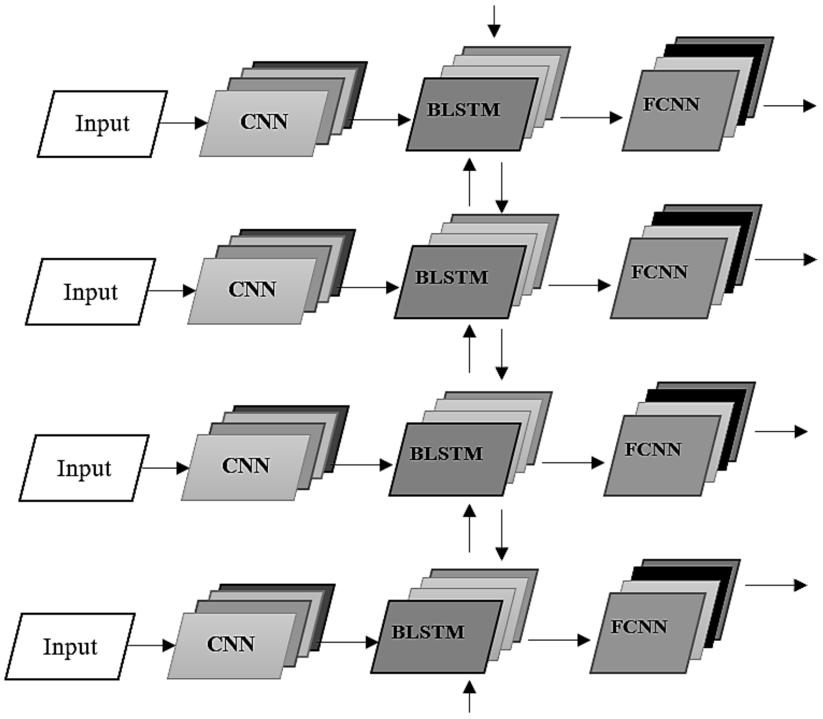

Fig. 3. Overall architecture of the proposed model

The final block in the proposed architecture is fully connected neural network block which is normal neural network. without neural network, SoftMax function can be used to predict the schedule as a probability function. The resource with low probability values are removed and high probability values are held in the scheduling process. while using fully connected network, instead of static threshold-based result prediction, the network model learns the information from output of previous blocks. Leveraging all the outputs from bidirectional RNN the performance of proposed model is improved due to the neural network learning procedure. The process is similar to image classification from the outputs of CNN blocks but here it needs to learn from the RNN block. Mathematically the network model is formulated for one layer with activation function is given as

$$\mathcal{A}^{[\mathcal{L}]} = \sigma(\mathcal{Z}^{[\mathcal{L}]}) = \sigma(w^{[\mathcal{L}]}\mathcal{A}^{[\mathcal{L}-1]} +^{[\mathcal{L}]}),$$

where the activation function is represented as $\mathcal{A}^{[\mathcal{L}]}$, weight function is represented as $w^{[\mathcal{L}]}$ and layer is represented as $\mathcal{L}$. The bias term for the layer $\mathcal{L}$ is represented as $\vartheta^{[\mathcal{L}]}$. The maximum probability obtained for the weight are formulated as

$$P(\mathcal{Y} = f \mid \varphi) = \frac{e^{\varphi} w_j}{\sum_{f=1}^{\mathcal{F}} e^{\varphi} w_j},$$

where the probability of prediction from SoftMax output is represented as $P(\mathcal{Y} = f \mid \varphi)$, φ represents the set input and w_j represents the weights. To train the fully connected neural network, cross entropy function is employed in the proposed work. The output of the cross-entropy function measures the error between predicted results and actual results. The major objective of the training process is to reduce the prediction error. The function is mathematically formulated as

$$\mathcal{J} = -\frac{1}{\vartheta}\sum_{i=1}^{\vartheta}(\mathcal{O}^{(i)} \log(\alpha^{[\mathcal{L}](i)})) + (1-\mathcal{O}^{(i)})\log(1-\alpha^{[\mathcal{L}](t)}),$$

where the desired output function is represented as $\mathcal{O}^{(i)}$, training samples are represented as ϑ and activation function for layer $\mathcal{L}$ is represented as $\alpha^{[\mathcal{L}](t)}$. The layer parameters are updated using Adam optimizer. The optimization function is mathematically expressed as

$$w^{[\ell]} = w^{[\ell]} - \delta \frac{\rho^t \partial w[\ell]}{\sqrt{\eta^t \partial w[\ell] + \xi}};$$

where
$$\rho^t \partial w[\ell] = \frac{\rho^{t-1}\partial w[\ell]}{1-(\beta_1)^{'}},$$

and
$$\rho^{t-1}\partial w[\ell] = \beta_1 \rho^{t-1}\partial w[\ell] + (1-\beta_1)\frac{\partial \mathcal{J}}{\partial w[\ell]};$$

$$\eta^t \partial w[\ell] = \frac{\eta^{t-1}\partial w[\ell]}{1-(\beta_1)^{'}},$$

and
$$\eta^{t-1}\partial w[\ell] = \beta_2 \eta^{t-1}\partial w[\ell] + (1-\beta_2)\frac{\partial \mathcal{J}}{\partial w[\ell]},$$

where β_1 and β_2 represents the weighted average functions, the learning rate is represented as δ, the squares of gradients before bias correction is represented as

η^{t-1}, η^{t} represents the after bias correction. In order to avoid zero ξ is added in the denominator. Similarly, the CNN blocks are also optimized in the sameman-ner as neural network block. Summarized pseudocode for the proposed resource scheduling model is presented as follows.

Pseudocode for the proposed deep bidirectional RNN based resource scheduling

```
Initialize learning procedure
Input: encoded data, network parameters
Output: optimal resource
While do
Forward model
       Load network parameters and data
       For t $\in$ time slots do
Initialize forward propagation from CNN
            Block LSTM to neural network  block
Store the prediction results
              Store the back propagation values
perform predicted results update
     Model backward
    For t$\in$ time slots do
Obtain parameter gradients in neural network block
Obtain LSTM block parameters
Obtain CNN block parameters
Compute the gradient and store
Perform update of parameters
          Update optimizer function
       End
   End
End
```

RESULTS AND DISCUSSION

The proposed resource scheduling algorithm using deep recurrent neural network and convolutional neural network performance is verified through simulation analysis and compared with existing resource scheduling techniques. Intel Berkeley research laboratory benchmark dataset has been used for experimentation. The dataset includes sensor readings from 54 sensors acquired from light, voltage, humidity and temperature sensors. The time duration for data collection is about 2 months and measurements are performed for every 31 seconds. Simulation analysis is performed in NetBeans version 8.1 installed in an Intel i5 processor with 16GB memory. Performance metrics like response time, execution time, resource utilization, efficiency is considered for scheduling model. Clustering accuracy and convergence rate are used for improved spectral clustering model performance analysis. As detailed analysis of spectral clustering performances are explored in the previous research work this experimental analysis mainly focused the performance of deep learning-based scheduling techniques. Table 1 depicts the details of clustering model performance over existing *k*-means and fuzzy *c*-means clustering performances.

T a b l e 1 . Performance analysis of clustering models

Algorithms	Convergence Rate, %	Clustering Accuracy, %
k-means	94.30	92.73
FCM	95.90	95.09
Improved spectral clustering algorithm	99.00	99.15

Fig. 4 depicts the performance comparative analysis of proposed model and existing scheduling methods like improved particle swarm optimization (IPSO), genetic algorithm (GA), LSTM based scheduling methods. The resource utilization is measured based on number of clusters and maximum resource utilization is obtained by the presented model on contrary to existing methodologies. The average resource utilization attained by the IPSO and GA models are 95.92% and 95.62% which is 4% lesser than the proposed scheduling model. Resource utilization attained by the LSTM based scheduling model is 98.68% which is lesser than the proposed deep BRNN model.

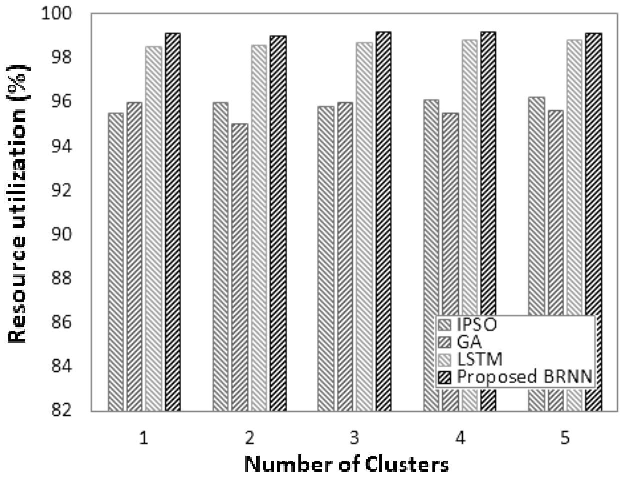

Fig. 4. Resource Utilization

The response time analysis is comparatively presented in Fig. 5 for the proposed model and existing models. The minimum time taken by the algorithm to schedule a resource is measured as response time. It is observed from the results if the number of clusters are minimum the response time of all the models is less whereas it gradually increases as the number of clusters increases. The average response acquired by the hybrid deep learning BRNN model is 1.54 seconds which is much better than the existing resource scheduling approaches. Further the performance of all the models are measured in terms of overall execution time. The process of requesting resources, time taken to check the resource availability, time acquired to schedule the resource are collectively measured as overall execution time.

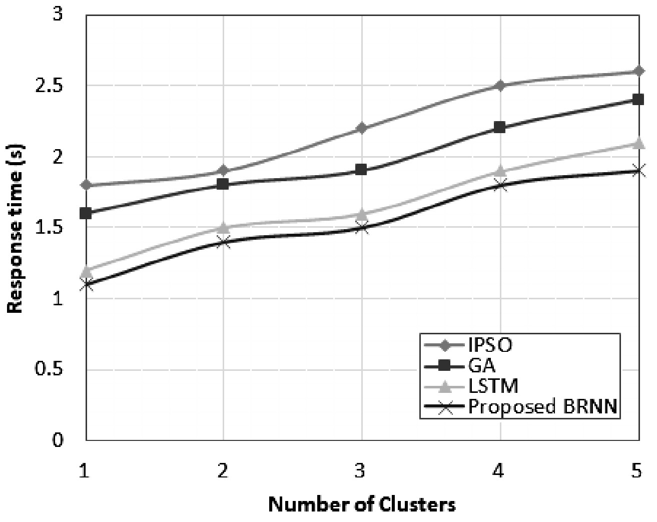

Fig. 5. Response Time Analysis

Fig. 6 depicts the comparative analysis of proposed model and existing models execution times with respect to number of iterations. The maximum iteration is selected into 50 and the execution time gradually increases for all the methods when iteration increases. The overall execution time attained by the proposed model is 14.82 seconds. Though the performance of existing methods are varied in seconds, the small time difference will introduce huge impact in service violations and affect the quality of services. Presented model schedules the resources effectively based on the prediction characteristics of bidirectional model which reduces the further analysis if same resource is requested in future. Whereas existing methods analyze the request again and confirms the resource status and schedule to the respective request increases the overall execution time.

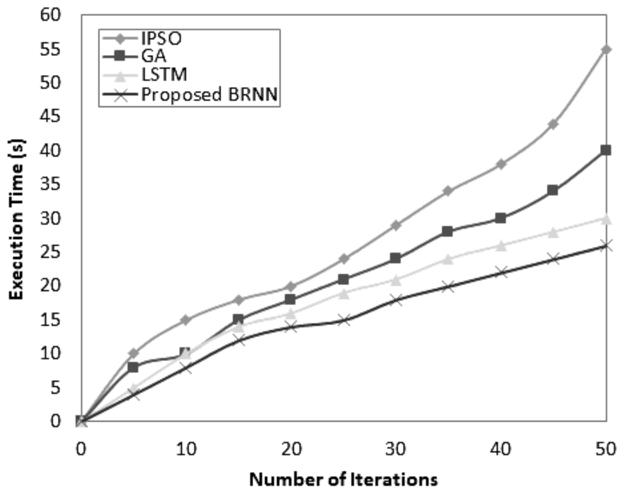

Fig. 6. Execution Time Analysis

The average delay exhibited by the existing models and proposed model is analyzed and depicted in Fig. 7. The time taken by a user to acquire an optimal resource is generally termed as waiting time. Whereas delay describes about the time which exceeds the stipulated predefined time period. Since all the requests are fixed with a minimum waiting time and meanwhile the system need to search for the resource and schedule them for further process. if the minimum waiting time is exceeding over a period and the resources are scheduled after that then it is measured as delay. Here the proposed model exhibits minimum delay compared to existing scheduling procedures. The prediction performance and memory utilization of proposed model not only reduces the execution time also it reduces the delay by schedule the resource based on existing utilization. Whereas there is no process followed in the existing scheduling procedures increases the delay.

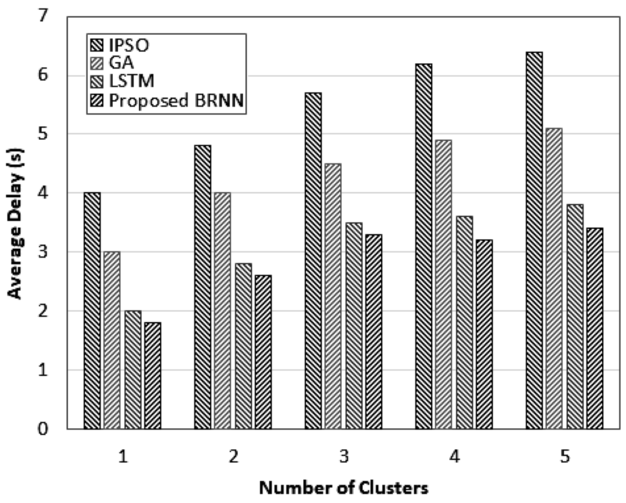

Fig. 7. Average Delay Analysis

The overall efficiency of all the approaches are comparatively analyzed and depicted in Fig. 8. Based on the response time, resource utilization, execution time and delay the efficiency is measured. If the algorithm exhibits maximum delay definitely it will introduce an impact in the efficiency. Similarly, if the resource utilization is low then that system could not be considered as an efficient one. It is essential for a system to complete the required process with minimum response time and execution time. Considering all these factors, the proposed model attains maximum efficiency score compared to other methods. Since the proposed scheduling procedure reduces the response time, execution time and improves the resource utilization and minimizes the delay which indicates the maximum efficiency.

T a b l e 2 . Performance Comparative Analysis

Methods	**Resource Utilization (%)**	**Response Time (s)**	**Execution Time (s)**	**Average Delay (s)**	**Efficiency (%)**
IPSO	95.92	2.20	26.09	5.42	94.50
GA	95.62	1.98	20.73	4.30	96.00
LSTM	98.68	1.66	17.55	3.14	98.00
Proposed BRNN	99.12	1.54	14.82	2.86	98.90

Table 2 depicts the summary of proposed model and existing resource scheduling models performances in terms of resource utilization, response time, execution time, average delay and efficiency. It can be observed from the results that the proposed model attains better performance than existing approaches. improved scheduling performance will increase the overall performance and quality of services in cloud integrated IoT networks.

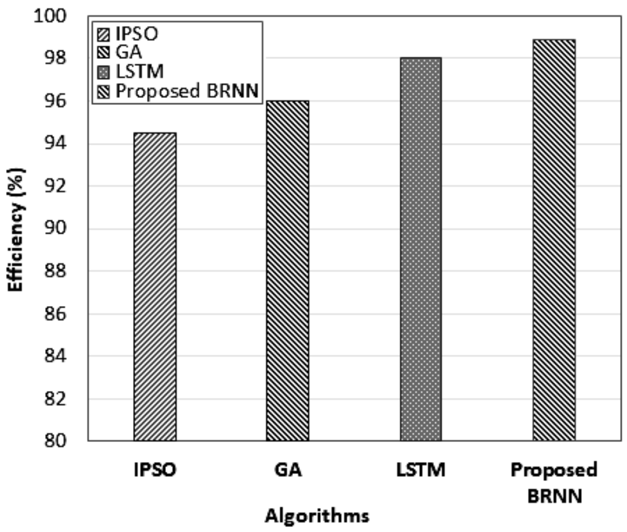

Fig. 8. Efficiency Analysis

CONCLUSION

A hybrid deep learning model for resource scheduling in edge computing Internet of Things (IoT) network is presented in this research work. The presented scheduling algorithm includes the deep bidirectional recurrent neural network with convolutional neural network block to improve the scheduling performance in edge computing. Initially the resource requests are clustered using improved spectral clustering algorithm and converted into encoded image. The encoded information features are processed by one-dimensional convolutional neural network model followed by bidirectional long-short term memory which is type of RNN model. The final results select the optimal resources and schedule to respective requests to reduce the computational complexity of IoT network. Simulation analysis of proposed model demonstrates the better performances compared to existing scheduling models which is based on improved particle swarm optimization algorithm, genetic algorithm and LSTM based model. Further, this research work can be extended by introducing concatenated deep learning techniques to avoid initial clustering process and improve the overall performances.

REFERENCES

1. Vishnu Kumar Kaliappan, S. Gnanamurthy, and K. Mohanasundaram, “Reduced power consumption by resource scheduling in mobile cloud using optimized neural network,” *Materials Today: Proceedings*, vol. 46, no.15, pp.6453–6458, 2021.
2. Yanling Shao, Chunlin Li, and Youlong Luo, “Cost-effective replication management and scheduling in edge computing,” *Journal of Network and Computer Applications*, vol.129, pp. 46–61, 2019.

3. Xiao Ma, Shangguang Wang, Shan Zhang, Peng Yang, Chuang Lin, and Xuemin Shen, "Cost-Efficient Resource Provisioning for Dynamic Requests in Cloud Assisted Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 968–980, 2021.
4. Yongsheng Hao, Jie Cao, and Jinglin Du, "Energy-aware scheduling in edge computing with a clustering method," *Future Generation Computer Systems*, vol.117, pp.259–272, 2021.
5. Jun Liu, Tianfu Yang, and Bo Sun, "Resource allocation and scheduling in the intelligent edge computing context," *Future Generation Computer Systems*, vol.121, pp. 48–53, 2021.
6. Quyuan Luo, Shihong Hu, Changle Li, Guanghui Li, and Weisong Shi, "Resource Scheduling in Edge Computing: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.
7. Xiaomin Li, Jiafu Wan, Hong-Ning Dai, Muhammad Imran, Min Xia, and Antonio Celesti, "A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.
8. Samrat Nath and Jingxian Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.
9. Lin Wang, Lei Jiao, Jun Li, Julien Gedeon, and Max Mühlhäuser, "MOERA: Mobility-Agnostic Online Resource Allocation for Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2019.
10. Fengjun Zhao, Ying Chen, Yongchao Zhang, Zhiyong Liu, and Xin Chen, "Dynamic Offloading and Resource Scheduling for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2154–2165, 2021.
11. Yuan Zhang, Peng Du, Jiang Wang, Teer Ba, Rui Ding, and Ning Xin, "Resource Scheduling for Delay Minimization in Multi-Server Cellular Edge Computing Systems," *IEEE Access*, vol. 7, pp. 86265–86273, 2019.
12. Bin Zhang and Dexu Chen, "Resource scheduling of green communication network for large sports events based on edge computing," *Computer Communications*, vol.159, pp. 299–309, 2020.
13. Qi Zhang, Lin Gui, Shichao Zhu, and Xiupu Lang, "Task Offloading and Resource Scheduling in Hybrid Edge-Cloud Networks," *IEEE Access*, vol. 9, pp. 85350–85366, 2021.
14. Hyame Assem Alameddine, Sanaa Sharafeddine, Samir Sebbah, Sara Ayoubi, and Chadi Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.
15. Tong Liu, Yameng Zhang, Yanmin Zhu, Weiqin Tong, and Yuanyuan Yang, "Online Computation Offloading and Resource Scheduling in Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6649–6664, 2021.
16. Umber Saleem, Yu Liu, Sobia Jangsher, Yong Li, and Tao Jiang "Mobility-Aware Joint Task Scheduling and Resource Allocation for Cooperative Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2021.
17. Jiaying Meng, Haisheng Tan, Xiang-Yang Li, Zhenhua Han, and Bojie Li, "Online Deadline-Aware Task Dispatching and Scheduling in Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1270–1286, 2020.
18. Haitao Yuan and MengChu Zhou, "Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, 2021.

19. Youngjin Kim, Chiwon Song, Hyuck Han, Hyungsoo Jung, and Sooyong Kang, "Collaborative Task Scheduling for IoT-Assisted Edge Computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020.
20. Yu Liu, Yong Li, Yong Niu, and Depeng Jin, "Joint Optimization of Path Planning and Resource Allocation in Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2129–2144, 2020.
21. Wenhan Zhan et al., "Deep-Reinforcement-Learning-Based Offloading Scheduling for Vehicular Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.
22. Yi-Han Chiang, Tianyu Zhang, and Yusheng Ji, "Joint Cotask-Aware Offloading and Scheduling in Mobile Edge Computing Systems," *IEEE Access*, vol. 7, pp. 105008–105018, 2019.
23. Shuaishuai Guo, Dalei Wu, Haixia Zhang, and Dongfeng Yuan, "Resource Modeling and Scheduling for Mobile Edge Computing: A Service Provider's Perspective," *IEEE Access*, vol. 6, pp. 35611–35623, 2018.
24. Tien Van Do, N.H. Do, and Peter Hegyi, "Comparison of scheduling algorithms for multiple mobile computing edge clouds," *Simulation Modelling Practice and Theory*, vol. 93, pp. 104–118, 2019.
25. G. Vijayasekaran and M. Duraipandian, *An Efficient Clustering and Deep Learning Based Resource Scheduling for Edge Computing to Integrate Cloud-IoT. Wireless Pers Commun*. 2022. Available: https://doi.org/10.1007/s11277-021-09442-8.

Resieved 00.00.2022

INFORMATION ON THE ARTICLE

G. Vijayasekaran, Department of Computer Science and Engineering, Sir Issac Newton College of Engineering and Technology, Nagapattinam, India, e-mail: gvijayasekaran2@gmail.com

M. Duraipandian, Department of Computer Science and Engineering, Hindusthan Institute of Technology, Coimbatore, India, e-mail: durainithi@gmail.com

ПЛАНУВАННЯ РЕСУРСІВ У МЕРЕЖАХ IOT EDGE COMPUTING З ВИКОРИСТАННЯМ ГІБРИДНОГО АЛГОРИТМУ ГЛИБОКОГО НАВЧАННЯ / Г. Віджаясекаран, М. Дурайпандіан

Анотація. Поширення Інтернету речей (IoT) і бездротових сенсорних мереж покращує передачу даних. Попит на передачу даних швидко зростає, що викликає появу парадигми периферійних обчислень. Граничні обчислення відіграють важливу роль у мережах IoT і надають обчислювальні ресурси поблизу користувачів. Перенесення служб із хмари до користувачів розширює комунікаційні, сховища та мережеві функції користувачів. Однак масивні мережі IoT потребують великого обсягу ресурсів для своїх обчислень. Щоб досягти цього, у граничних обчисленнях використовуються алгоритми планування ресурсів. Алгоритми планування ресурсів, засновані на статистиці та машинному навчанні, розвинулися протягом останнього десятиліття, але їх продуктивність можна покращити, якщо додатково проаналізувати вимоги до ресурсів. У роботі подано глибоке планування ресурсів на основі навчання в периферійних обчислювальних мережах IoT з використанням глибокої двонаправленої рекурентної нейронної мережі (BRNN) і алгоритмів згорткової нейронної мережі. Перед плануванням користувачі IoT класифікуються в різні кластери за допомогою спектрального алгоритму кластеризації. Пропонований аналіз моделювання перевіряє продуктивність з точки зору затримки, часу відгуку, часу виконання та використання ресурсів. Існуючі алгоритми планування ресурсів, якот генетичний алгоритм (GA), покращена оптимізація роїв частинок (IPSO) і моделі на основі LSTM, порівнюються із запропонованою моделлю для підтвердження кращої продуктивності.

Ключові слова: периферійні обчислення, хмарні обчислення, інтернет речей (IoT), планування ресурсів, глибоке навчання.