Head Node Selection Algorithm in Cloud Computing Data Center

Samira Kanwal, Zeshan Iqbal, Aun Irtaza, Muhammad Sajid, Sohaib Manzoor, and Nouman Ali

1Department of Computer Science, University of Engineering and Technology Taxila, Taxila 47050, Pakistan
2Department of Electrical Engineering, Mirpur University of Science and Technology (MUST), Mirpur-AJK 10250, Pakistan
3Department of Software Engineering, Mirpur University of Science and Technology (MUST), Mirpur-AJK 10250, Pakistan

Correspondence should be addressed to Samira Kanwal; samira.kanwal@uettaxila.edu.pk

Received 24 July 2020; Revised 14 June 2021; Accepted 28 June 2021; Published 26 July 2021

1.Introduction

The recent advancements in Internet technologies and cloud computing have transformed the way data is processed. Cloud computing provides an environment including different resources, for example, memory, storage, and processing cores, for hosting applications and processing data according to the user requirement. It offers users flexible, pay-per-use, and on-demand scalable services hosted at a remote data center which can be accessed from anywhere and at anytime. The data center is comprised of a large number of networked servers and nodes that work together in a cluster to share resources. Cloud computing systems rely heavily on nodes coordination for the parallel execution of the tasks. A cluster leader node manages the task of communication and synchronization among the other nodes. Consequently, the selection of an optimal leader node is an ultimate requirement for preventing the network from becoming unstable. The leader election to choose the best node in a data center is a challenging problem because leaders are responsible for managing segregated data, sharing resources among nodes, and overcoming latency. Hence, communication between nodes is impossible without the leader/master nodes, so leader selection must be carefully considered. As the nodes are directly linked to the leader, the leader ensures that
there will never be a deadlock among the network nodes and that tasks will be processed efficiently.

For an effective leader election process in distributed networks and cloud computing environments, the research community has proposed several protocols [1–8] and algorithms (Biswas et al.) [9–16]. A bully algorithm is proposed in [17–20] for leader election which selects the leader nodes dynamically based on node ID criteria. In [9, 21, 22], the authors proposed a ring algorithm, in which each node shares its ID with all other nodes and keeps track of them in a database. Based on priority, the algorithm selects one node as a leader from this list. In [11, 23–25], node IDs are generated at random, and each node is assigned a priority number. After that, the node with the highest priority number is selected as the leader node. For leader election in [26–28], a message-passing strategy has been proposed. The comparison of these recent techniques that are based on ID and are randomly generated is shown in Table 1. However, these recent techniques failed to consider the maximum resources of the hosts due to increasing the complexity. These approaches introduce latency because of the higher message passing rate and slow response of the nodes. Some recent works related to leader election in IoT are presented in [29–31]. The major drawback of the above-mentioned methods is that the node’s resource profile is never considered. As a result, selecting the weaker leader node becomes equally likely. In high-load situations, the weak leader crashes, necessitating reelection of the leader, which slows down overall processing and delays task execution across network nodes. As a result, node resources must be considered during the leader election phase to ensure that task processing occurs efficiently and the network is stable. In this paper, the Head Node Selection Algorithm (HNSA) is proposed to select a leader node in the cloud data center. The HNSA is based on a genetic algorithm (GA). The proposed HNSA selects the node as a head that has more resources than other nodes. The proposed algorithm selects the head node to ensure effective job execution, resource sharing, and communication among the nodes without any delay and in effective manner. Our proposed method gets the metainformation about jobs, VMs, and available hosts in the data centers and computes the fitness value of different hosts that are based on available host resources such as CPU-MIPS, throughput, bandwidth, and RAM. Using the metainformation, generate the initial population of chromosomes for the operation of GA. The chromosome comprises multiple jobs on different hosts, and it is taken as a gene. The chromosome length depends on the total number of jobs that are submitted in the cluster of data centers. To select the head node from each cluster, we locate that host which has the maximum fittest value against the fitness function. This fitness value is calculated after performing the GA operations, that is, crossover and mutation. In our proposed method, the crossover scheme used the swapping technique among the jobs on hosts, whereas, in mutation swapping, the one individual means to swap the job on the host. The fitness value depends on resources like CPU-MIPS, RAM, bandwidth, and throughput; these resources are the factors of our fitness function. This proposed approach is significant to utilize in a unified way to select the head node efficiently. It successfully selects the head node in the cluster for efficient job execution and reduces any delay among the nodes. The proposed technique also considers the sudden failure of the head node. When the head node is failed due to some reason like the host overloaded within the time interval, we select the candidate node as a head node for smooth communication and execution of jobs. So, our approach’s performance is efficient without suffering from any vulnerability like communication delay or failure of the host. To justify and validate the impact of our approach, we perform some experiments and compare our approach against some state-of-the-art algorithms. The following are the main contributions of this research:

(i) A novel method is proposed by using a genetic algorithm-(GA-) based Head Node Selection Algorithm (HNSA) for efficient data processing in the data center of cloud computing.

(ii) The proposed model provides efficient data processing and resource sharing due to optimal head node and candidate node selection through the HNSA.

(iii) Our proposed technique is efficient in SLA violation, execution time, energy consumption, and memory utilization as compared to state-of-the-art approaches.

The remaining part of the paper is structured as follows: In Section 2, the state of the art in head node selection is presented. In Section 3, the proposed GA-based head node selection methodology is explained. In Section 4, an experiment is conducted and the outcomes are presented; and, finally, we conclude our work in Section 5.

2. Related Work

In this section, we have reviewed the existing techniques related to frequent leader/head node selection. Ktari et al. [40] presented an agent-based selection approach employing dynamic trees. The main concern of the authors was to
maintain a forest tree in which the root node was selected as the leader which was determined by the highest ID value. In the process of leader election, the ID value was created randomly without taking into account any other resources. Similarly, in [41], a method for leader selection based on the probabilistic investigation of traffic lights was presented.

In [42], an approach, namely, old algorithm of the ring, was utilized, which employed the unidirectional interface to join all hubs or nodes. This technique was concerned to ensure that a job was successfully running on all nodes. In this paper, the system worked by randomly producing a distinctive priority number and a node was selected as the leader that had the highest priority. The failure of the leader hub caused the entire procedure to be performed again to find out the new leader. The leader hub selection approach needed about \(2(n - 1)\) messages, communicated all over the network in which initially \((n - 1)\) messages were transmitted to initiate the leader selection procedure. Finally, again \((n - 1)\) messages were passed to nominate the novel leader. Later on, in [43], EffatParvar et al. had tried to enhance the traditional ring algorithm mentioned in [42] by considering the requirement relations together with the verification of numerous distributed protocols employed for leader nomination. The approach in [43] utilized the Temporal Ordering Specification Language (TOSL) along with Analysis of Distributed Processes (ADP).

Several changes were introduced in the existing approach, that is, introduction of the bully algorithm [44], which was based on a tree structure and performed the comparison message complexity. Similarly, in [26], an extended form of bully leader selection method was presented in the environment of cloud computing in which the leader node selection was based on a Super Node (SN) [45] improving the leader nomination speed and message complexity was minimized to \(O(n^2/k)\) to \(O(n^2)\), in which \(k\) was showing the total nodes, while \(k\) was presenting the district. Moreover, the approach in [43] had the leader election complexity of \(O(k^2n^2)\), where \(n\) was presenting the total nodes and \(k\) was an upper bound value on the multiplicity of the labels. Similarly, in [46], a methodology for nominating the leader node was presented. The major motivation of this work was the selection of a single leader node; therefore, the failure of the leader caused the entire network to fail, which resulted in restarting the entire procedure. The approach in [47] had the election complexity of \(O(\log n)\) and the message complexity in this [24] is \(O(k^2n^2)\).

To deal with the problem of single leader election, two approaches have been presented in [9], which worked with nominating multiple leaders to avoid the risk of communication delays and minimized the latency rate. Three empirical techniques were employed in a cohesive way for calculating the proper status of leaders in a polynomial period. Another model for leader node selection was introduced in [48] based on employing the probabilistic grounded framework. The work had presented the improvements on the energy consumption and the consistency problem of channel communication. In literature, few approaches have been concerned with the leader selection in a peer-to-peer and distributed network [49, 50]. The method in [49] proposed an approach for the peer-to-peer network, while for distributed environment a technique was introduced in [51] employing software mediators to improve the speed of leader election procedure and minimize the energy consumption. Moreover, some protocols for fault-tolerance-based leader election in asynchronous distributed systems were presented in [52]. In [22, 53], two new mobility conscious methods for leader election were presented in the environment of ad hoc network systems. These techniques ensured that each connected node in the topology had been associated with a single leader. The approaches in [22, 54–56] have highly relied on a temporarily ordered routing technique called TORA.

After performing the analysis of existing techniques, we have analyzed that most of techniques in literature for selecting a leader are highly dependent on a priority number or on a unique identifier that is usually randomly generated by a framework. These approaches do not take into account the network topology and availability of real-time resources before performing the leader node selection procedure [7, 57–60]. Moreover, with the increase in the complexity of the cloud computing environment, the leader election task becomes more challenging. To deal with the mentioned problem, several techniques have been presented; however, still these frameworks do not take into account all resources because of increased computational cost. We are presenting a novel approach for the leader/head node selection which is based on the idea of GA.

3. Proposed Research Model

The presented work comprises three main modules. The first is “initial population generation,” the second is “head node selection,” and the third is “candidate node selection” for efficient network performance. Consider a data center \(D_c\) with multiple servers called hosts \(H = \{h_1, h_2, h_3, \ldots, h_i\}\) that are grouped in the form of \(C = \{C_1, C_2, C_3, \ldots, C_p\}\) and each host \(h_i\) has \(R_m = \{R_{hi1}, R_{hi2}, R_{hi3}, \ldots, R_{him}\}\) resources, for example, bandwidth, CPU, throughput, and memory. In the data center, jobs like \(J = \{j_0, j_1, j_2, \ldots, j_{h1}, j_k\}\) that contain multiple tasks to each cluster \(C_p\) are submitted. These jobs or tasks are randomly mapped on different hosts \(H_i\) that can make chromosomes sets like \(X = \{x_1, x_2, x_3, \ldots, x_k\}\), where \(x_i = \{h_{ij1}, h_{ij2}, h_{ij3}, h_{ij4}, \ldots, h_{ijk}\}\) and \(h_{ijk}\) represents genes in a chromosome. Apply the genetic operations, that is, crossover and mutation, on the chromosomes that are generated by using the initial population \(P = \{p_1, p_2, p_3, \ldots, p_n\}\). Afterward, the fitness function is applied against each chromosome for evaluation. Correspondingly, the head nodes are selected from the evaluated chromosome with the highest value. The responsibility of head nodes is to allocate the jobs to each node in the cluster to effectively improve the efficiency load balancing of the network. The structural design of the HNSA is presented in Figure 1.
3.1. Initial Population Generation. In our proposed algorithm, generate the initial population by mapping the jobs on different hosts \( h_i \) by using the scheduling algorithm. There are many ways to schedule the jobs between the intermediate nodes. In the proposed method, initial population formation is scheduled as follows: \( j_1 \) on \( h_2 \), \( j_2 \) on \( h_1 \), \( j_3 \) on \( h_i \), and \( j_k \) on \( h_3 \), as illustrated in Figure 2. There are several numbers of chromosomes in each set of the population represented as \( P_q = X_1, X_2, X_3, ..., X_n \). Each set of chromosomes contains multiple genes, and the genes represent map jobs on different hosts. For example, the chromosome \( X_1 \) is represented as \( X_1 = \{ h_1 j_2, h_2 j_1, h_1 j_3, ..., h_{(i-1)j_{(k-1)}} h_i j_k \} \).

3.2. Head Node Selection. In this phase, we perform some genetic operations like crossover and mutation. In the crossover procedure, select two random population individuals called chromosomes. There are multiple crossover strategies; we use two cut-point techniques because we select two random points for cut by using the following equation:

\[
[\beta_1, \beta_2] = \text{rand}(2) \ast (k - 1) + 1,
\]

where the random number is generated by using \( \text{rand}(2)(kl) + 1 \), for selecting the two cut-point positions \( \beta_1 \) and \( \beta_2 \). These two cut-point positions are selected in two sets of chromosomes which have the highest fitness values. When selecting two random cut points \( \beta_1 \) and \( \beta_2 \), perform the crossover operation. The crossover operation is performed from two new individuals that are called offspring in the population, as shown in Figure 3. In this way, better offspring are generated for the next population, and repeat this process till the best solution is achieved.

After the crossover, we perform the mutation process on the offspring that comes from the crossover operation. This mutation process is unary that applies to the single feature of offspring. During the mutation, process deviations are very minimal because generally a very small value is selected. In our method, the substitution process is used in mutation. So, we select any random position from the offspring and substitute this value as shown in Figure 4.

Now, we evaluate the excellence of our proposed method by using the fitness function \( F_c \). In \( F_c \), we calculate the fittest value of all solutions through the following equation:

\[
F_c = \max \left[ \sum_{i=0}^{N} f_h \right],
\]

where \( f_h \) is the fitness of each host which is calculated by using the following equation:

\[
f_h = \Theta_1 C_a + \Theta_2 R_a + \Theta_3 B_a + \Theta_4 T_a,
\]

where the system parameters like CPU-MIPS, memory, bandwidth, and throughput are considered. The values of these parameters are altered by considering SLA (Service Level Agreement).
Level Agreement) that works on the base of knapsack algorithm [61], shown in Table 2. The parameters with higher values of weight will impact the fitness value more as compared to the parameters with fewer values. In our proposed method, we used tournament selection, selecting the individual or the host which has the highest fitness value that is calculated by using the fitness function from a randomly generated population, which will be selected as a head node and candidate node.

The objective of our proposed method is to choose the leader node or head node with the higher (α) resources.

### 3.3. Selection of Candidate Node

After using GA operation, we will select a head node in each cluster. In this phase, choose a candidate node as a head node from each cluster C in a data center. The reason for choosing the candidate node as a head is because the head node is failed before the time interval and then the candidate becomes the head node. The candidate node is the backup of the head node and takes all responsibilities of the head node. The proposed technique will return two optimal hosts from each cluster, and we will choose one as a head node and the other as a candidate node. After each time interval, our system will run to optimize the head node and candidate node (Algorithm 1).

### 4. Results and Discussion

A few experiments are performed to examine the performance and efficiency of the proposed method by comparing it against some state-of-the-art algorithms. The details of results and discussion are mentioned in the following subsections.

#### 4.1. Experimental Setup

To evaluate the efficiency and performance of the proposed method, an extensive set of experiments are performed by using CloudSim Plus [20] simulator. The CloudSim simulator is based on CloudSim [21]; it is a framework for designing and simulating the data centers. CloudSim Plus simulates the more realistic scenarios because it is an extension of CloudSim. For experimental simulation, numerous sets of heterogeneous hosts/servers (MIPS range: 1000–4000) are used. The processor AMD-Ryzen 5 2500 is used for processing and the Python library is used to create the graph. The parameters execution time, energy consumption, SLA violation, and memory utilization are used to evaluate the performance of the proposed method. The time in which the host finishes the job is called execution time. Energy consumption is how much energy is utilized during the job execution. The memory utilization is how much MIPS are allocated to the host during the interval of time. When the job is not fully executed in the given time frame, the SLA violation occurs. To evaluate and analyze the efficiency of the proposed method, the graphs are generated by using input metrics as used in [62].

#### 4.2. Experimental Analysis

In this section, we describe the particular facts about how to experiment. Experiments were used to equate the performance efficiency of the proposed method with some state-of-the-art algorithms. In all experiments, system parameters like RAM, CPU-MIPS, throughput, and bandwidth were used. The system parameter weightage is given according to the SLA, means the user’s requirements, as shown in Table 3. The following experiments are performed as explained below.
4.2.1. Evaluation through Host Increment. In the first conducted experiment, the proposed method (HNSA) is compared with some state-of-the-art algorithms in terms of effectiveness. The performance evaluation is based on parameters like execution time, energy consumption, memory utilization, and SLA violation. In this experiment setup, the performance is evaluated by increasing the number of hosts/servers. However, a fixed number of tasks are used in the whole scenario. The host range is 20–100 along with the increment of 20, and the range of the tasks is fixed to 1000. In terms of execution time, the performance of the proposed HNSA is more efficient than those of state-of-the-art algorithms as shown in Figure 5. During the performance analysis, the proposed HNSA takes minimum time to completely execute a job as compared to state-of-the-art algorithms like BLA and HEFT. Figures 6 and 7 show that the proposed algorithm also performs better in terms of SLA violation and memory utilization. The utilization of memory is improved because the number of hosts is increased and the number of tasks is fixed. The first conducted experiment’s results validate that the proposed HNSA had a better performance and improved the efficiency in terms of execution time, memory utilization, energy consumption, and SLA violation.

4.2.2. Evaluation through the Task/Job Increment. The second conducted experiment was to calculate the performance efficiency of the proposed method with the same parameters. In this experiment, the number of tasks is incrementing; however, a fixed number of hosts is used. The tasks’ range is from 200 to 20,000 with an increment of 200 and the number of hosts is 50, which is fixed during the whole simulation. Figures 8 and 9 show that the performance of the proposed method is efficient in terms of SLA violation and execution time. In this, the experimental results validate the performance efficiency based on parameters like execution time and SLA violation with the increment of tasks. State-of-the-art algorithms like BLA and HEFT had inefficient performance in terms of execution time and SLA violation as compared to HNSA. The results of this second experiment validate that the proposed algorithm takes less time to complete the job execution. Also, it validates that the HNSA reduces the SLA violation as compared to BLA and HEFT algorithms during the workflow execution.

4.2.3. Evaluation through Different Weightages (Θ). In this experiment, different weightages are given to parameters like CPU-MIPS, memory, bandwidth, and throughput to analyze the performance efficiency of HNSA against the BLA and HEFT algorithms. During this simulation, we used two

| Input: List of Jobs and Servers |
|--------------------------------|
| Output: Head Node, Candidate Node |
| 1) Population = randomly generate; |
| 2) while i = 1 to total population |
| 3) population list = randomly assign each job to a host |
| 4) end while |
| 5) calculate the cost (); |
| 6) if time = scheduling interval |
| 7) for (! Stopping criteria) |
| 8) Perform Crossover Function(); |
| 9) Perform Mutation Function(); |
| 10) Calculate Fitness Function (); |
| 11) end for |
| 12) leader and candidate node = host with the maximal value from equation (1) in the first index of the population list is head node and the second is candidate |
| 13) if (Head Node Failed) |
| 14) candidate = host at the second index of the population list became a head node |
| 15) end if |
| 16) end if |

Algorithm 1: HNS.
different scenarios and gave different weightages to parameters according to the user requirement. In scenario 1, the range of hosts is fixed as 50 and the number of tasks/jobs is increasing by 1000. However, the weightages of resources like RAM, throughput, bandwidth, and CPU are different. The parameter weightages in scenario 1 are CPU 10%, RAM 60%, bandwidth 20%, and throughput 10%, as shown in Table 4. Figure 10 validates that the performance of HNSA is better compared to state-of-the-art algorithms in terms of execution time and SLA violation. In scenario 2, the numbers of jobs and hosts are the same during the evaluation of the performance of HNSA against those of some state-of-the-art algorithms using all parameters. However, the weightages of all parameters are different: CPU-MIPS 10%, bandwidth 70%, RAM 10%, and throughput 10% as scenario 1, shown in Table 4. Figure 11 validates the simulation results in scenario 2, where the performance of the HNSA is more efficient compared to the BLA and HEFT algorithms. The second conducted experiment was to calculate the performance efficiency of the proposed method with the same parameters. In this experiment, the number of

| Metrics | Θ |
|---------|---|
| CM      | 0.5 |
| M       | 0.3 |
| Bd      | 0.1 |
| Th      | 0.1 |

Table 3: System metrics weights.
tasks is incrementing; however, the number of hosts used is fixed. The range of tasks is from 200 to 20,000 with an increment of 200 and the number of hosts is 50, which is fixed during the whole simulation. Figures 10 and 11 show that the proposed method’s performance is efficient in terms of SLA violation and execution time. In this, the experimental results validate the performance efficiency based on parameters like execution time and SLA violation with the increment of tasks. State-of-the-art algorithms like BLA and HEFT had inefficient performance in terms of execution time and SLA violation as compared to HNSA. The results of this second experiment validate that the proposed algorithm takes less time to complete the job execution. It also validates that the HNSA reduces the SLA violation as compared to BLA and HEFT algorithms during the workflow execution.

4.2.4. Scheduling Comparison of Jobs. In this experiment, we analyze the performance of the HNSA in comparison to some state-of-the-art algorithms such as HEFT and BLA. In this simulation, ten tasks/jobs are randomly generated, and the length of each task/job is as follows: 61000, 62000, 63000, 64000, 65000, 66000, 67000, 68000, 69000, and 70000. Table 5 shows the execution time of each job.

Figure 12 shows the execution time with state-of-the-art algorithms and Figure 13 shows the execution time of tasks on different servers/hosts using the proposed algorithm. The HNSA maps the tasks on those hosts which efficiently meet the job/task requirement and are executed efficiently. Through HNSA, T9 runs on h2/S2 with minimum time as compared to BLA compiling T9 on S3, so HNSA efficiently executes the jobs as compared to BLA.

4.2.5. Energy Consumption with the Increment of Tasks and Hosts. In this simulation, the energy consumption is evaluated with two different scenarios. In the first scenario, energy consumption is calculated through the increment of the hosts, and in the second scenario, it is calculated through increment of the number of jobs. Figures 14 and 15 show the energy consumption of HNSA and state-of-the-art algorithms in kw/h of each job to be executed. The experiment results of this first scenario show that the HNSA reduces the energy consumption due to efficient host uses for job execution. The optimized resources of the efficient host are used to execute the job in minimum time and other servers/hosts turn off to save energy. So, the HNSA algorithm uses optimized hosts to utilize the maximum resources and turn off the unused hosts and map the tasks/jobs on available optimal hosts.
5. Conclusion

In our methodology, we focused on the problem of the head node selection in the cloud data center. In the data center, the leader node is to ensure efficient execution of the task/jobs and deliver the well-organized services to the users according to their requirements. So, the selection of a head/leader node is a challenging task. However, in literature, most of the techniques are proposed to overcome the issues of head node selection. Still, there is a problem with all proposed techniques because these techniques failed to consider the resources of the host/server. So, we proposed an optimized algorithm called the HNSA, inspired by the processes of the genetic algorithm. Our proposed algorithm is based on a random population approach, which combined the behaviors of individual mean jobs on hosts. Through the HNSA, select the head node and candidate node based on available resources. The reason for the candidate node selection is that if the head node is failed within the time interval, then the candidate node becomes head node without long delay. To validate the performance efficiency and reliability of the proposed HNSA, we performed some simulations and compared these results against those of some state-of-the-art algorithms. The performance evaluation parameters are execution time, SLA violation, memory utilization, and energy consumption. Our results validate that our proposed HNSA is more efficient compared to state-of-the-art algorithms. In the future, we will implement new protocols, and this technique will also be implemented in IoT environment, wireless network, and fog computing.
Data Availability

Data sharing is not applicable to this article because the authors have used data that are generated randomly, the details of which are in the section titled “experiments and results” within this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Subbiah, P. Varalakshmi, R. Prarthana, and C. R. Devi, “Energy efficient big data infrastructure management in geo-federated cloud data centers,” Procedia Computer Science, vol. 58, pp. 151–157, 2015.
[2] T. J. Bittman, G. J. Weiss, M. A. Margevicius, and P. Dawson, Magic Quadrant for X86 Server Virtualization Infrastructure, Gartner, Stamford, CT, USA.
[3] R. van Renesse, R. Rodrigues, M. Spreitzer, C. Stewart, D. Terry, and F. Travostino, “Challenges facing tomorrow’s datacenter: summary of the LADS workshop,” in Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, pp. 1–7, New York, NY, USA, September 2008.
[4] S. M. Shetty and S. Shetty, “Analysis of load balancing in cloud data centers,” Journal of Ambient Intelligence and Humanized Computing, pp. 1–9, 2019.
[5] H.-S. Chiang, M.-Y. Chen, and Y.-J. Huang, “Wavelet-based EEG processing for epilepsy detection using fuzzy entropy and associative petri net,” IEEE Access, vol. 7, pp. 103255–103262, 2019.
[6] J. A. Meda-Campaña, “On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs,” IEEE Access, vol. 6, pp. 31968–31973, 2018.
[7] G. Xia, Y. Huang, F. Li et al., “A thermally flexible and multi-site tactile sensor for remote 3D dynamic sensing imaging,” Frontiers of Chemical Science and Engineering, vol. 14, no. 6, pp. 1039–1051, 2020.
[8] G. Hernández, E. Zamora, H. Sossa, G. Téllez, and F. Furlán, “Hybrid neural networks for big data classification,” Neurocomputing, vol. 390, pp. 327–340, 2020.
[9] T. Biswas, R. Bhardwaj, A. K. Ray, and P. Kuila, “A novel leader election algorithm based on resources for ring networks,” International Journal of Communication Systems, vol. 31, no. 10, Article ID e3583, 2018.
[10] M. Kanwa, P. Sharma, and G. Mehrrota, “Novel leader election algorithm using buffer,” in Proceedings of the 2017 2nd International Conference on Telecommunication and Networks (TET-NET), pp. 1–4, IEEE, Noida, India, August 2017.
[11] R. Ingraham, T. Radeva, P. Shields, S. Viqar, J. E. Walter, and J. L. Welch, “A leader election algorithm for dynamic networks with causal clocks,” Distributed Computing, vol. 26, no. 2, pp. 75–97, 2013.
[12] G. Aquino, J. D. J. Rubio, J. Pacheco et al., “Novel nonlinear hypothesis for the delta parallel robot modeling,” IEEE Access, vol. 8, pp. 46324–46334, 2020.
[13] J. de Jesús Rubio, “SOFMfLS: online self-organizing fuzzy modified least-squares network,” IEEE Transactions on Fuzzy Systems, vol. 17, no. 6, pp. 1296–1309, 2009.
[14] I. Elias, J. D. J. Rubio, D. I. Martinez et al., “Genetic algorithm with radial basis mapping network for the electricity consumption modeling,” Applied Sciences, vol. 10, no. 12, p. 4239, 2020.
[15] Y. Zhou, Y. Huang, J. Pang, and K. Wang, “Remaining useful life prediction for supercapacitor based on long short-term memory neural network,” Journal of Power Sources, vol. 440, Article ID 227149, 2019.
[16] Y. Zhou, Y. Wang, K. Wang et al., “Hybrid genetic algorithm method for efficient and robust evaluation of remaining useful life of supercapacitors,” Applied Energy, vol. 260, Article ID 114169, 2020.
[17] K. Shukla and B. Sahoo, “Dynamic selection of the best server out of multiple servers available using bully algorithm in any geographical area,” in Advances in Electrical Control and Signal Systems, pp. 871–877, Springer, Berlin, Germany, 2020.
[18] Q. Eh Samul, K. Mamun, and S. M. Masum, “Modified bully algorithm for electing coordinator in distributed systems,” 2004.
[19] S. S. Supase and R. B. Ingle, “Are coordinator election algorithms in distributed systems vulnerable?” in Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–5, IEEE, Kharagpur, India, July 2020.
[20] P. K. Thakur, R. Kumar, R. Ali, and R. Kumar Malviya, “A new approach of bully election algorithm for distributed computing,” International Journal of Electrical, Electronics and Computer Engineering (IJEECE), vol. 1, no. 1, pp. 72–79, 2011.
[21] E. R. S. Subramanian, B. Sri Gurubaran, A. S. Sayee Shruthi, V. Ashwarya, N. Balaji, and A. Umanakeswari, “A novel hypergraph-based leader election algorithm for distributed systems,” in Innovations in Computer Science and Engineering, pp. 437–445, Springer, Berlin, Germany, 2020.
[22] N. Malpani, J. L. Welch, and N. Vaidya, “Leader election algorithms for mobile ad hoc networks,” in Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 96–103, Boston, MA, USA, August 2000.
[23] L. Du and Z. Yu, “Algorithm for distributed super-node election,” Jisuanji Gongcheng Yu Yingyong (Computer Engineering) vol. 6, pp. 31968–31973, 2018.
[24] S. Tucci-Piergiovanni and R. Baldoni, “Eventual leader election algorithm for mobile ad hoc networks,” in Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, pp. 31973–31978, Springer, Berlin, Germany, 2020.
[25] A. D. Durand, “Élections et anneaux unidirectionnels en présence d’homonymes,” in Proceedings of the ALGOTEL 2017-19èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, La Rochelle, France, September 2017.
[26] C. Maurer, “Leader election algorithms,” in Nonsynchrounous Distributed Computing and Networks (ICDCN), pp. 381–395, Springer, Berlin, Germany, 2021.
[27] R. Ingram, M. Radeva, P. Shields, J. Walter, and J. Welch, “An asynchronous leader election algorithm for mobile networks without perfect clocks,” 2009.
[28] S. Tucci-Piergiovanni and R. Baldoni, “Eventual leader election in infinite arrival message-passing system model with bounded concurrency,” in Proceedings of the 2010 European Dependable Computing Conference, pp. 127–134, IEEE, Valencia, Spain, April 2010.
[29] P. K. R. Maddikunta, T. R. Gadekallu, R. Kaluri, G. Srivastava, R. M. Parizi, and M. S. Khan, “Green communication in IoT networks using a hybrid optimization algorithm,” Computer Communications, vol. 159, pp. 97–107, 2020.
[30] M. Alazab, K. Lakshmanna, G. T. Reddy, Q.-V. Pham, and P. K. Reddy Maddikunta, “Multi-objective cluster head
selection using fitness averaged rider optimization algorithm for IoT networks in smart cities," *Sustainable Energy Technologies and Assessments*, vol. 43, Article ID 100973, 2021.

[31] N. Kadjojou, A. Bounceur, M. Bezoui et al., “A dominating tree based leader election algorithm for smart cities IoT infrastructure,” *Mobile Networks and Applications*, pp. 1–14, 2020.

[32] D. Becker, F. Junqueira, and M. Serafini, “Leader election for replicated services using application scores,” in *Proceedings of the ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 289–308, Springer, Lisbon, Portugal, December 2011.

[33] A. Mostafa and A. E. Youssef, “A leader replacement protocol based on early discovery of battery power failure in MANETs,” in *Proceedings of the 2013 International Conference on IT Convergence and Security (ICITCS)*, pp. 1–4, IEEE, Macau, China, December 2013.

[34] C. Suvarna Lakshmi and M. R. Murthy, “A hybrid of improved bulls and weighted round robin to optimize the leader and load balancing in cloud and distributed computing environment,” *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 1, pp. 9–16, 2018.

[35] M. G. Murshed and A. R. Allen, “Enhanced bully algorithm for leader node election in synchronous distributed systems,” *Computers*, vol. 1, no. 1, pp. 3–23, 2012.

[36] J. Gu, Y. Zhou, W. Wu, and T. Chen, “Analyzing eventual leader election protocols for dynamic systems by probabilistic model checking,” in *Proceedings of the International Conference on Cloud Computing and Security*, pp. 192–205, Springer, New York City, NY, USA, July 2015.

[37] K. Altsen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore, “Leader election in asymmetric labeled unidirectional rings,” in *Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 182–191, IEEE, Orlando, FL, USA, May 2017.

[38] A. Favier, N. Guittoneau, L. Arantes, A. Fladenmuller, J. Lejeune, and P. Sens, “Topology aware leader election algorithm for dynamic networks,” in *Proceedings of the 2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 1–10, IEEE, Perth, Australia, December 2020.

[39] B. Sidik, R. Puzis, P. Zilberman, and Y. Elovici, “PALE: partially asynchronous agile leader election,” 2018, https://arxiv.org/abs/1801.03734.

[40] M. Ktari, M. A. Haddar, M. Mosbah, and A. H. Kacem, “Distributed computation and maintenance of a spanning tree in dynamic networks by mobile agents,” in *Proceedings of the 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 331–336, IEEE, Crans-Montana, Switzerland, March 2016.

[41] N. Fathollahnejad, R. Barbosa, and J. Karlsson, “A probabilistic analysis of a leader election protocol for virtual traffic lights,” in *Proceedings of the 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 311–320, IEEE, Christchurch, New Zealand, January 2017.

[42] H. Garcia-Molina, “Elections in a distributed computing system,” *IEEE Transactions on Computers*, vol. C-31, no. 1, pp. 48–59, 1982.

[43] M. EftatParvar, N. Yazdani, M. EftatParvar, A. Dadlani, and A. Khonsari, “Improved algorithms for leader election in distributed systems,” in *Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology*, pp. V2–V6, IEEE, Chengdu, China, April 2010.

[44] B. Johansson, M. Rågberger, A. V. Papadopoulos, and T. Nolte, “Heartbeat bully: failure detection and redundancy role selection for network-centric controller,” in *Proceedings of the IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2126–2133, IEEE, Singapore, October 2020.

[45] J. Augustine, T. Kulkarni, P. Nakhe, and P. Robinson, “Robust leader election in a fast-changing world,” 2013, https://arxiv.org/abs/1310.4908.

[46] K. Altsen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore, “Election in unidirectional rings with homonyms,” *Journal of Parallel and Distributed Computing*, vol. 146, pp. 79–95, 2020.

[47] S. Sivasubramaniam, T. Kulkarni, and J. Augustine, “Leader election in sparse dynamic networks with churn,” *Internet Mathematics*, vol. 12, no. 6, p. 1611, 2016.

[48] Y. Sudo, F. Ooshita, T. Izumi, H. Kakugawa, and T. Masuzawa, “Time-optimal leader election in population protocols,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2620–2632, 2020.

[49] C. Newport, “Leader election in a smartphone peer-to-peer network,” in *Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 172–181, IEEE, Orlando, FL, USA, May 2017.

[50] P. Chaparala, A. R. Atmakuri, and S. S. S. Rao, “3-phase leader election algorithm for distributed systems,” in *Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 898–904, IEEE, Erode, India, March 2019.

[51] E. Amiri, H. Keshavarz, A. S. Fahleyani, H. Moradzadeh, and S. Komaki, “New algorithm for leader election in distributed WSN with software agents,” in *Proceedings of the 2013 IEEE International Conference on Space Science and Communication (IconSpace)*, pp. 290–295, IEEE, Melaka, Malaysia, July 2013.

[52] S.-H. Park, S.-G. Lee, and S.-C. Yu, “Notice of Violation of IEEE Publication Principles A fault tolerant election protocol in asynchronous distributed systems with fail-stop model,” in *Proceedings of the 2015 12th International Conference on Information Technology-New Generations*, pp. 44–48, IEEE, Las Vegas, NV, USA, April 2015.

[53] P. Karunakaran, H. Bagheri, and M. Katz, “Energy efficient multicast data delivery using cooperative mobile clouds,” in *Proceedings of the 18th European Wireless Conference 2012, European Wireless 2012 (VDE)*, pp. 1–5, Poznan, Poland, April 2012.

[54] K. Wang, L. Li, Y. Lan, P. Dong, and G. Xia, “Application research of chaotic carrier frequency modulation technology in two-stage matrix converter,” *Mathematical Problems in Engineering*, vol. 2019, Article ID 2614327, 8 pages, 2019.

[55] K. Wang, L. Li, T. Zhang, and Z. Liu, “Nitrogen-doped graphene for supercapacitor with long-term electrochemical stability,” *Energy*, vol. 70, pp. 612–617, 2014.

[56] N. Parajuli, A. Alsadoon, P. Prasad, R. S. Ali, and O. H. Alsadoon, “A recent review and a taxonomy for multimedia application in mobile cloud computing based energy efficient transmission,” *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 31567–31594, 2020.

[57] A. R. Molla and D. Shur, “Smoothed analysis of leader election protocols for network-centric controller,” in *Proceedings of the International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pp. 183–198, Springer, Austin, TX, USA, November 2020.
D. Fajardo-Delgado, J. A. Fernández-zepeda, and A. G. Bourgeois, “Randomized self-stabilizing leader election in preference-based anonymous trees,” *International Journal of Foundations of Computer Science*, vol. 23, no. 4, pp. 853–875, 2012.

M. Fardad, F. Lin, and M. R. Jovanović, “Algorithms for leader selection in large dynamical networks: noise-free leaders,” in *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7188–7193, IEEE, Orlando, FL, USA, December 2011.

K. Fitch and N. E. Leonard, “Information centrality and optimal leader selection in noisy networks,” in *Proceedings of the 52nd IEEE Conference on Decision and Control*, pp. 7510–7515, IEEE, Firenze, Italy, December 2013.

F.-T. Lin, “Solving the knapsack problem with imprecise weight coefficients using genetic algorithms,” *European Journal of Operational Research*, vol. 185, no. 1, pp. 133–145, 2008.

S. Kanwal, Z. Iqbal, A. Irtaza, R. Ali, and K. Siddique, “A genetic based leader election algorithm for IoT cloud data processing,” *Computers, Materials & Continua*, vol. 68, no. 2, pp. 2469–2486, 2021.