Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 22(1) March 2021

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 22, Number 1, March 2021

TABLE OF CONTENTS

Regular Papers:


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 1–13, DOI 10.12694:/scpe.v22i1.1808

A NOVEL APPROACH FOR CLUSTER HEAD SELECTION USING TRUST FUNCTION IN WSN

VIPUL NARAYAN* AND A.K. DANIEL†

Abstract. The enhancement of new technology in the sensor network shows a significant result in every aspect of life such as military surveillance, hospitals, mining and hospitals etc. The nodes are scattered randomly in RoI (Region of Interest) and data is transmitted to Base Station (BS) using the multi-hop technique. The Wireless Sensor Network (WSN) become an important research field for challenging problems as energy consumption, efficient cluster head selection process, routing algorithm, network strength, packet loss, energy loss and so forth. The agenda in the paper is to enhance Residual Energy (RE) of nodes and network lifetime. The problem is solved by using an efficient clustering and Cluster Head (CH) selection process.The cluster head selection is based on the maximum node residual energy and the minimum distance from the base station. The Proposed protocol worked in two stages. The new Threshold value T(H) is calculated for the cluster head selection process in the first stage. The data fusion method based on the trust function is used to get accurate data in the second stage. The energy model is utilized to reduce the excessive energy transmission inside the network. The Proposed protocol is compared with Stable Election Protocol and achieves 44% lifetime improvement, 59% stability improvement and 15% in survival rate respectively.

Key words: Clustering, Cluster Head Selection, Energy Efficiency, Wireless Sensor Network, Trust.

AMS subject classifications. 68M14

1. Introduction. Recent growth in the field of the mobile internet and sensor network technology has motivated the mindset of people towards new technology [1][2]. The Sensor Nodes (SN) are distributed to cover the network area and interact with the external environment using certain communication factors. The SN have limited battery power. The multiple SN are required to perform real-time tracking of objects in the network. In the twenty-one century, the WSN has played a significant impact on human lives. The evolution of science engineering and advanced technology in many disciplines improve sensor nodes technology and influences daily life nowadays. In year 1970s, the first generation sensor network was introduced. The SN follow point to point communication to transfer the data to BS. The next generation sensor network instead comprises less energy and operated independently with the cooperation of former nodes and collects data in the network. In the early 1990s, the third generation sensor network was introduced in the market with advance features and use a device manager and bus connection system to collect the information in the network. The fourth generation sensor networks have an advanced characteristic which performs a multi-hop approach and self-organizing features in the network for data transmission in the WSN [3][4]. The sensor network collects the information from various sources and sent to BS via single/multiple hop fashion. The sensor objects and observers play a significant role together for establishing communication via the communication link in the network. The multiple SN are deployed in the critical zone area after regular interval of time because data collection in these places are very difficult. The location of every SN in the network is not easily tracked. The GPS is used in some sensor nodes to know the exact location in the network. The sensor network has played an important role in the current scenario as deployement strategy,protocol design and green communication etc.[5]. The various applications are handled remotely and embedded therefore,requirements of sensor network in every place for the scientific and research purpose [6]. The modern sensor network technology is much more efficient than the traditional network. The node energy and environmental interruption affects the communication in the network. The

*Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur, India. (vipulupsainian2470@gmail.com).

†Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur, India. (danielak@rediffmail.com).

effective route optimization technique and utilization of resources in the network solve the problem which maximizes bandwidth utilization and maintain QoS in the WSN. The nodes are placed at static location in the hazardous region and sometimes unable to cover the network area due to several obstructions. The problem is solved by using energy efficient-coverage protocol and improves network lifetime. The RSSI is used to determine the current power of the signal. The complete coverage and optimum utilization of energy while transmitting the data in in the network become important research in the present scenario [7][8]. There are two types of transmission in the sensor network. The first is the actual transmission and another is the data fusion rate. In actual transmission the data is lost due to environment factors and data corruption. The data fusion rate helps in node computing, storage computing and reduce the consumption of energy inside the WSN. The nodes are randomly distributed in network and every time long distance node use more RE compared to shortest distance.

The paper proposed a protocol that uses new Threshold value for advanced nodes and normal nodes to prevent randonmness in the selection process of CH. The new Threshold value consits of weighted energy and distance ratio to prevent CH(s) and low node failure. The data fusion method based on trust function to get accurate data and energy model are utilized to reduce the energy transmission in the WSN.

The remaining section is defined as follows. The Design issues and Related work are discussed in section 2. The LEACH protocol in section 3. The Proposed Protocol is presented in section 4. The Experimental results are performed and contrasted with SEP in section 5. Conclusion and future work describe in section 6.

2. Design Issues and Related Work. The important task in the sensor network is to enhance the RE of the SN and WSN life by using optimum energy conservation techniques. The WSN area is partitioned into the number of regions by using longitudinal distance to the BS. In [9] the regions constitute CH for transmitting data to BS via the multihop scheme. In [10] performed static clustering to reduce the overhead in the system. The node having maximum RE is used for the selection of CH in the WSN. The increased number of CH(s) in the region enhanced the RE of SN and lifetime of system [11]. In [12] proposed LEACH protocol and performed static clustering for the CH selection by utilizing maximum RE of SN and minimum distance from BS as parameter in the network. In [13] proposed a minimum spanning tree algorithm for the selection of CH and transmission of data to BS using a single-hop scheme in the network. In [14] introduced the LEACH protocol for an efficient cluster head selection process and used K means clustering method for proficient utilization of clusters node inside the network and improved the performance of the WSN. In [15] proposed a clustering technique for the organization of SN and efficient routing scheme for transmission of data to BS. The WSN is partitioned into different regions and CH is based on the SN having maximum RE and minimum distance from BS which enhances the network lifetime. In [16]used distance,energy and node density as a parameter to increase the lifespan of WSN. In [17] energy problem in LEACH protocol is solved by adjusting the CH formula bassed on distance and energy as important parameters in the network. In [18] used radius and weight function to elect candidate CH in the network area and transmission of data to BS via a multi-hop approach which minimizes the energy consumption in the network. In [19] introduced the K- Medoids algorithm for the selection of clusters inside the network which enhanced the lifespan of the WSN. In [20] introduced a new methodology that uses the fusion method based on the principle of ingredient analysis algorithms to minimize the energy issue in the network. In [21] authors said that unequal nodes energy consumption and CH selection process is randomly performed in the network. The problem is solved by an efficient CH selection process which enhanced the node's RE and the lifetime of the WSN. In [19] proposed a protocol in which WSN is partitioned into equal size regions and static clustering scheme is used to avoid the overhead problem and multi-hop scheme for transfer the data to the BS. In [22] proposed a protocol in which the CH and non-CH are selected according to the RE of SN. The maximum value is chosen as CH among them which reduced the consumption of energy inside the network and improve system performance. The CH selection in the LEACH protocol is not appropriate so causes various problems.

In [23] introduced trust model for secure data transmission and minimize various issues in the network. In [24] the trust model is used to secure various layers of communication in the network. In [25] proposed data fusion process relying on the degree of trust to enhanced the system performance. In [26] proposed the data fusion method based on multivariate streams of data to detect and avoid outliers in the system. In [27] proposed a Trust-Distrust protocol and use four stages as topology management, linq quality appraisal, grading

and secure data transmission based on grade points in the network for data routing to the BS. In [28] achieved secure communication based on trust function and energy efficient clustering algorithm in the network. In [29] proposed a protocol for the selection of CH based on trust function. The data fusion and trust function is utilized to avoid unnecessary transmission and achieve a higher packet delivery ratio in the system.

3. Low Energy Adaptive Cluster Hierarchy (LEACH). The LEACH is the first clustering protocol.The selection of CH in the protocol is performed randomly and average energy is distributed among all nodes to avoid energy issues and network survival period. In the protocol different clusters are formed and among them, one is elected as CH. The rest non-CH(s) nodes sent data to each respective CH(s) to avoid the redundancy problem occurred in the data. The CH(s) aggregate the data and sent it to BS. The non-CH(s) have the cluster header information and a small routing table is maintained by the CHs [22]. The unnecessary consumption inside the network is avoided by using a routing table in the protocol. The protocol has many advantages as well as disadvantages too. In LEACH protocol when cluster formation takes place, the energy is optimized in the stabilization phase. The random numbers are allocated as 0 and 1 to every SN and when the resulting number is greater compared to the threshold set T(n) will be chosen as CH for that round. The clustering process has two phases known as the establishing phase and stabilization phase. The formation of clusters takes place and message-id is broadcast in the network using signal strength and with the help of non-CH(s) nodes in the network. The CH(s) request all messages and the routing table is maintained and follow the TDMA schedule for all clusters. The data aggregation is performed on the basis of the routing table and finally, the data is sent to BS [30][31].

4. Proposed Model. The SEP protocol was introduced to minimize energy issues in the network [32][33]. The SEP protocol uses weighted probability for CH election in normal as well as advance nodes. The SEP protocol does not ensure whether the nodes are effectively deployed or not in the network.

This paper proposes a new protocol in which WSN is partitioned into regions and contains two different SN. The nodes for high energy are known as advanced nodes and for low energy nodes called normal nodes. The proposed protocol is executed in two phases. In the first phase, distribution of SN and new Threshold value for cluster head selection in the WSN. In the next phase the accuracy of data is preserved by utilizing the data fusion method based on trust function to obtained accurate data. The energy model is used to reduce the unnecessary energy transmission in the WSN. The proposed protocol uses new a cluster head selection mechanism which avoids randomness for CH selection process. The selection of CH(s) is performed by using maximum RE of SN and minimum BS distance. The new T(H) consists of distance ratio and weighted energy which avoids energy failure problems in the CH(s) and low node energy problem. Through this way,it enhances the CH(s) nodes RE. The CH(s) wait for the completion of the data communication task of their cluster nodes. The proposed protocol reduces the energy consumption in WSN and improves the system performance.

4.1. Network Assumptions for Designing the Proposed Protocol. The following presumptions are considered for designing the proposed protocol.

- The SN are scattered in a random fashion in the RoI.
- The BS has a continuous power supply and the SN are deterministic in nature.
- The RSSI Plays a major role in the distance estimation between two nodes.
- The battery is not rechargeable.

4.2. Energy Model. The Energy model for the proposed protocol shown in the Fig 4.1. For communicating c bits over distance d, the transmission energy (T_E) is required as follows [16]:

$$T_E = \begin{cases} c \times E_{elec} + c \times E_{fs} \times d^2, & \text{if d} < d_0 \\ c \times E_{elec} + c \times E_{mp} \times d^4, & \text{else} \end{cases} \tag{4.1}$$

where, E_{elec}= Electrical energy (E_{elec} required for the conversion of 1 bit of data to signal), E_{fs}= Power amount for free space, E_{mp}= Power amount for multipath models.

$$d_0 = \sqrt{\frac{E_{fs}}{E_{mp}}} \tag{4.2}$$

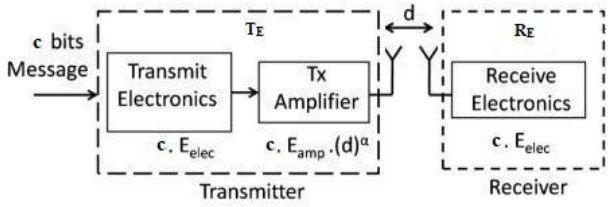


Fig. 4.1: Energy Model

The energy required for receiving c bits is as follows:

$$E_{RX} = c \times E_{elec} \tag{4.3}$$

The energy required by the Cluster Member(CM_E) for communicating c bits is as follows:

$$CM_E = c \times E_{elec} + c \times E_{fs} \times d_{CH} \tag{4.4}$$

where d_{CH}= distance to CH.

The energy required by the CH (CH_E) is calculated as follows:

$$CH_E = c \times m(E_{elec} + E_{fs} \times d_B + E_{DA}) \tag{4.5}$$

where m =count of cluster member's, d_B= distance from Base Station, E_{DA} = Aggregation Energy.

4.3. Trust Function. Let us assume that the set of SN as S = s_1, s_2,..., s_n is covering the entire network area. The d_i and d_j is the data collected by the sensor node s_i and s_j at same moment. If the accuracy of sensor data d_i is higher, than the trust degree of d_i is greater than the rest of the sensor data. When data d_i is trusted by d_j, than d_i covers all the possible degree for sensor data. Thus the trust degree function is represented as [25]:

$$t_{ij} = f(|(d_i - d_j)|) \tag{4.6}$$

where $t_{ij} = f(|(d_i - d_j)|) \in [0, 1]$ and $i, j = 1, 2, \ldots, n$. Based on the trust function t_{ij}, the trust matrix corresponding to n number of sensor nodes computes all same parameters used at the same time as follow:

$$T = \begin{bmatrix} t_{11} & \cdots & t_{1n} \\ \vdots & \ddots & \vdots \\ t_{n1} & \cdots & t_{nn} \end{bmatrix} \tag{4.7}$$

The weight of data (d_i) collected from the sensor node (s_i) is b_i. In the matrix T, t_{ij} represents the trust degree of d_i to d_j, but does not show the trust degree of all data with respect to d_i. The actual value of d_i is reflected by t_{i1}, t_{i2},...,t_{in}. Therefore, weight matrix B combine all the actual value of T with a set of the non-negative matrix A with values a_1, a_2, ...,a_n.

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = T \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \tag{4.8}$$

$$b_i = a_1 t_{i1} + a_2 t_{i2} + \ldots + a_n t_{in} \tag{4.9}$$

$$b_i = \sum_{j=1}^{n} a_j t_{ij} \tag{4.10}$$

The matrix B and A are scalar multiples, if $B = TA = \lambda A$. Here, λ is the Eigenvalue for the corresponding eigenvector A. Thus the degree of trust is measured as:

$$\frac{b_i}{b_j} = \frac{a_i}{a_j}, i, j = 1, 2, \ldots, n \tag{4.11}$$

The normalized b_i is:

$$b_i = \frac{a_i}{a_1 + a_2 + \ldots + a_n} \tag{4.12}$$

The data fusion is the process to collects data from multiple sources and gives meaningful information that is not performed by any single sensor node. The objective of data fusion is to enhance the QoS and achieves reliable and accurate data transmission in the RoI. The data fusion process also minimized data redundancy problems and minimized the energy conniption in the WSN. The data fusion expression is:

$$D_F = \sum_{i=1}^{n} b_i d_i \tag{4.13}$$

From equation 4.13 data fusion is also expressed as:

$$D_F = \frac{\sum_{i=1}^{n} a_i d_i}{a_1 + a_2 + \ldots + a_n} \tag{4.14}$$

4.4. Cluster Head Selection Phase. The CH are selected by using maximum nodes RE and minimum distance from BS.The nodes are categorized into two parts.The first is normal nodes and the second is advanced nodes.

$$N_P = \frac{P}{1 + N_n N_a} \tag{4.15}$$

$$A_P = \frac{P}{1 + N_n N_a} (1 + N_n) \tag{4.16}$$

where, N_P =Selection probability of normal node as CH, A_P = Selection probability of advanced node as CH, N_a = Advanced Node percentage, N_n = Amount of energy higher than the normal node.

The threshold is calculated as shown below:

$$T(H) = \begin{cases} \frac{P}{1 - P \times (r \mod \frac{1}{P})}, & \text{if n} \in \text{G} \\ 0, & \text{otherwise} \end{cases} \tag{4.17}$$

where $T(H)$ =Threshold Value, r =current number of round, P =The desired percentage of a node to be CH.

The new $T(H)$ is calculated by modifying in equation 4.17. The weighted energy (E) and distance ratio (D) are calculated as follow:

$$E = (RE_{current} - (E_a + E_t + E_r)) \tag{4.18}$$

$$D = \frac{D_{BS}}{D_L} \tag{4.19}$$

where $RE_{current}$ is current Residual Energy, E_a is Aggregation Energy, E_t is Transmission Energy, E_r is Reception Energy and D_{BS} is the distance from the BS, D_L is the longest distance from BS.The new $T(H)$ is

used to avoid the low energy problem and improves CH survival rate. By utilizing equation 4.15, 4.16, 4.17, 4.18, 4.19 simultaneously improves in distribution of Normal and Advanced nodes. The new $T(H)$ for Normal and Advanced nodes are given in equation 4.20 and 4.21 below.

$$T(H) = \begin{cases} \frac{N_{PX}}{1-N_P \times (r \mod \frac{1}{N_P})}(a \times E + b \times D), & \text{if n} \in \text{G} \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

$$T(H) = \begin{cases} \frac{A_{PX}}{1-A_P \times (r \mod \frac{1}{A_P})}(a \times E + b \times D), & \text{if n} \in \text{G} \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

where N_{PX} represents N_P with weight parameter X, A_{PX} represents A_P with weight parameter X, and a, b are the coefficient i.e. $a \in [0,1], b \in [0,1]$ and $a + b = 1$.

Equation 4.20 and 4.21 improves energy consumption in the WSN. The long-distance nodes consume much more energy compares to short distance nodes. Therefore by using new $T(H)$ for normal and advanced nodes improves the distribution of SN and enhanced the node RE in the WSN.

4.5. Proposed Protocol. The flow chart for the Proposed protocol is shown in Fig 4.2.

The network consist of advanced nodes and normal nodes. The new threshold value minimizes the randomness in the CH selection process. The data fusion rate is used to get accurate data using trust function. The energy model are used to avoid unnecessary energy transmission in the network.

Algorithm 1: The Data Prediction Phase

Initialization: H_E = Node High Energy, H = Transformed High Energy, L = Transformed Low Energy, CH = Cluster Head, BS = Base Station ;
$Node_i \leftarrow$ rand (0,1) ;
if $Node_i > H_E$ **then**
  Set H_$Node_i \leftarrow$ High Energy Node ;
else
  Set L_$Node_i \leftarrow$ Low Energy Node ;
Calculate energy radio weight ;
Calculate distance parameters ;
Transform H_$Node_i \leftarrow$ H ;
Transform L_$Nodei \leftarrow$ L ;
if *((H < T(H) && L < T(H))* **then**
  Broadcast $\leftarrow$ CH message ;
  Nodes receive $\leftarrow$ message ;
else
  Non_CH nodes $\leftarrow$ wait till message broadcast;
Nodes receive information intensity ;
Nodes receive message to join CH ;
CH receive request message ;
CH set TDMA schedule ;
Broadcast TDMA schedule ;
CH receive data ;
CH Fused data ;
CH send data to BS ;

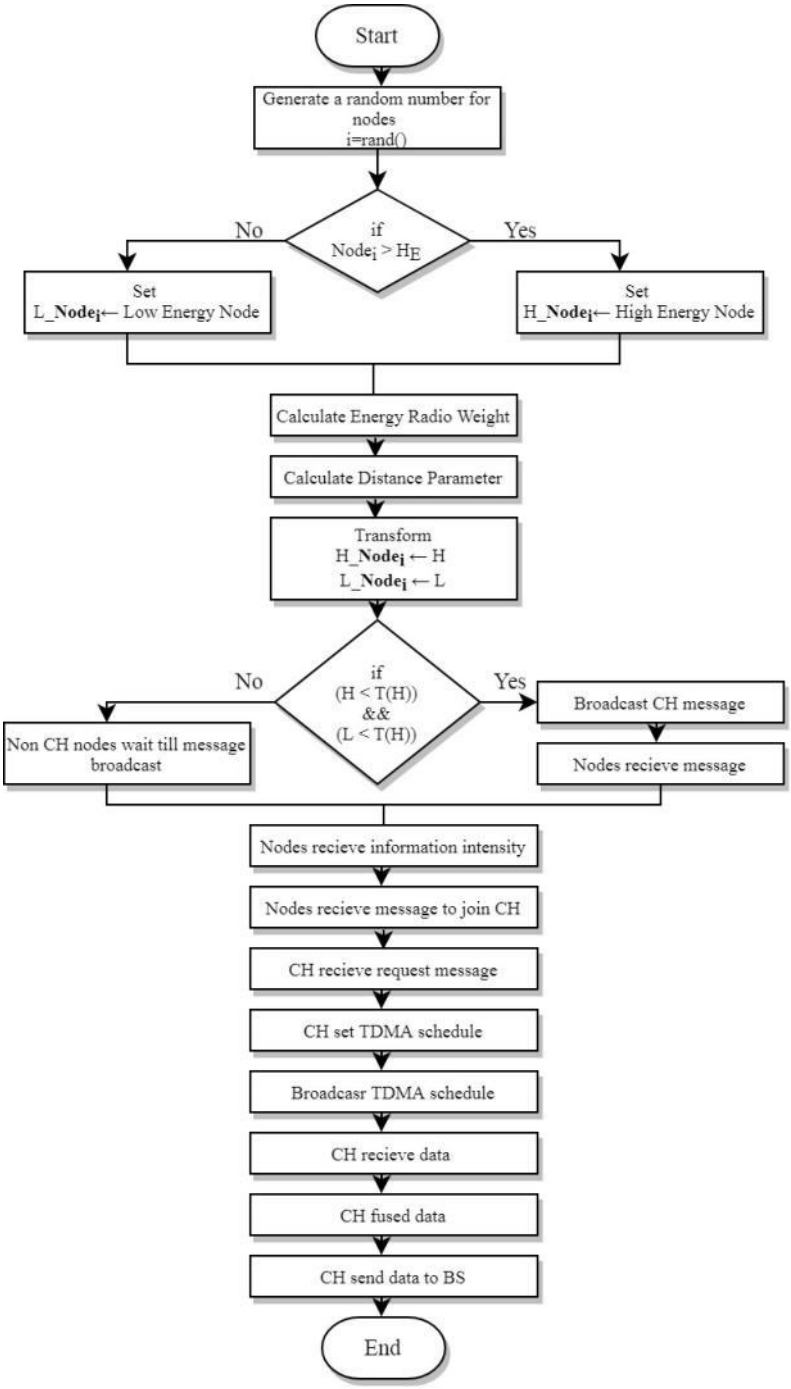


Fig. 4.2: Flow Chart of Proposed Protocol

Table 5.1: Parameter for Simulation

Parameters Used	Values
Nodes (N)	150
Network Area	(150,150)
Position of BS	(150,75)
E_{fs}	10 pJ/bit/m^2
E_{mp}	0.0013 pJ/bit/m^4
(E_0)	0.5 J
(E_{RX})	50 nJ/bit
(E_{DA})	5 nJ/bit/signal

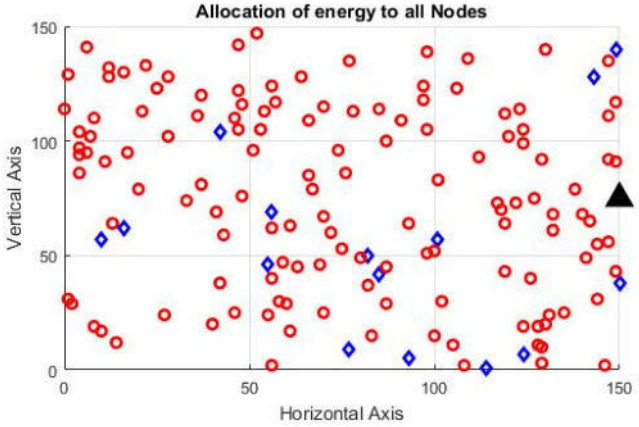


Fig. 5.1: The deployment of Normal Nodes and Advanced Nodes

5. Simulation Results And Execution Assessments. The simulation is performed in Matlab 2017a. The RoI is (150×150) m^2 area and the distribution of nodes are deterministic in the area. The probability of CH selection is 10%. The proposed protocol is compared with the SEP based on simulation parameter [34] as shown in Table 5.1.

The initial network deployment is as shown in the Figure 5.1.

In the Figure 5.1 the blue color represents the advanced nodes and the red color represents the normal nodes. The BS is located at (150, 75) on the X axis and the Y axis respectively and represented with black color. The effectiveness of the proposed protocol is compared with respect to the SEP for alive nodes is shown in Figure 5.2, dead nodes in Figure 5.3, number of packet transmitted to BS in Figure 5.4, nodes elected as CH in Figure 5.5, percentage of energy consumed in Figure 5.6, percentage of remaining energy in Figure 5.7 and survival rate percentage in Figure 5.8 respectively.

The results in Figure 5.2 shown that the proposed protocol has 29 alive nodes after the first iteration which is better than SEP.

In the Figure 5.3 shown the proposed protocol reveals better performance than SEP protocol because in the proposed protocol first node dies at 998 rounds where as in SEP protocol dies at 411 rounds.

As shown in the Figure 5.4, the packets transmitted to BS are 1.8×10^4 for the proposed protocol while in SEP protocol the value 1.7×10^4 has used for packet transmission.

As shown in Figure 5.5, the proposed protocol has more number of nodes as CH compares to the SEP protocol. It can be observed from the figure that the number of CH starts decreasing after 400 rounds in SEP and for the proposed protocol number of CH starts decreasing after 1000 rounds. So we can say that the stability period is higher for the proposed protocol compared with SEP.

In the Figure 5.6 shown that the energy consumption is reduced for the proposed protocol compared with

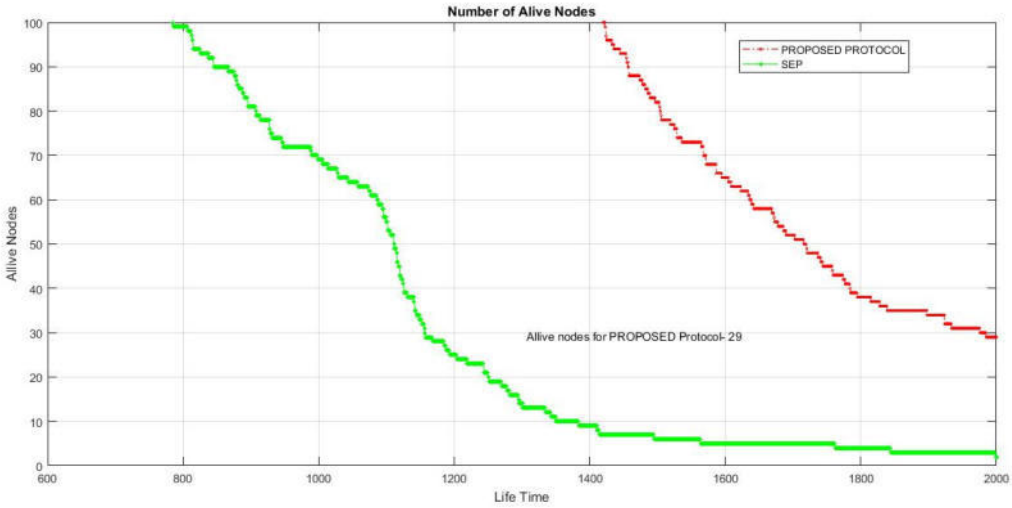


Fig. 5.2: Number of Alive Nodes

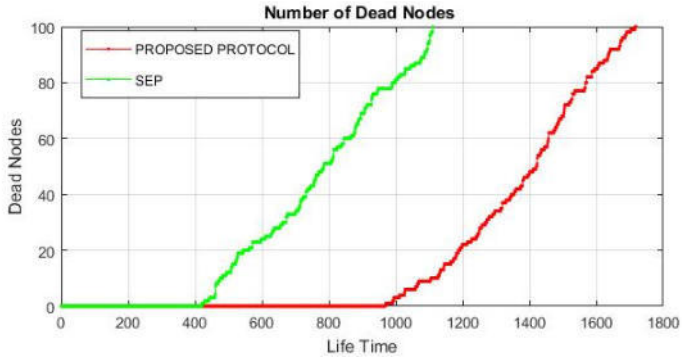


Fig. 5.3: Number of Dead Nodes

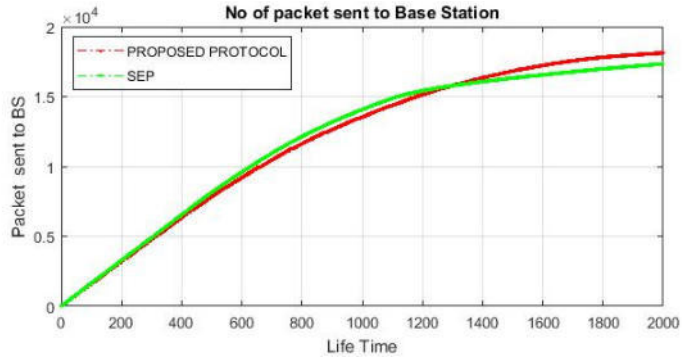


Fig. 5.4: Number of Packet Transmission to BS

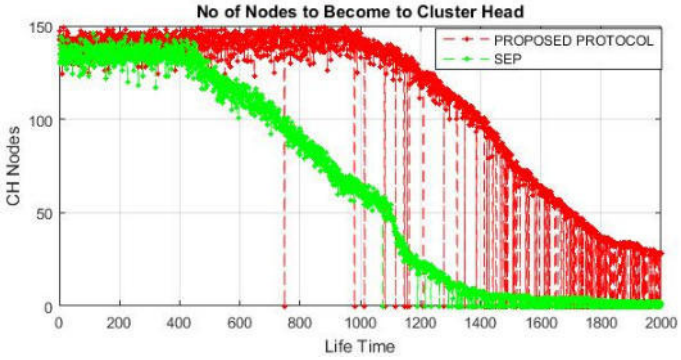


Fig. 5.5: Number of Node to become CH

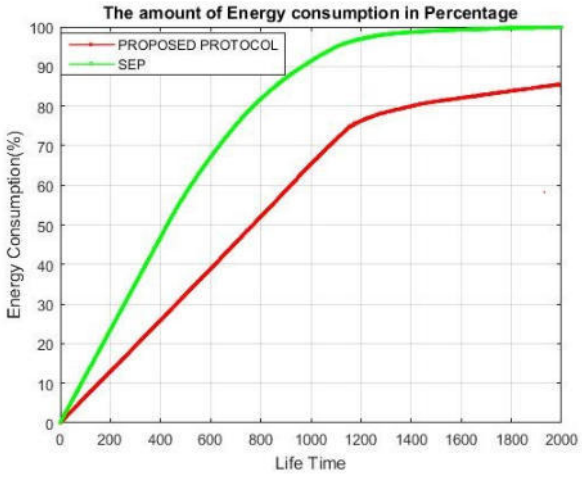


Fig. 5.6: The amount of energy consumption (in %)

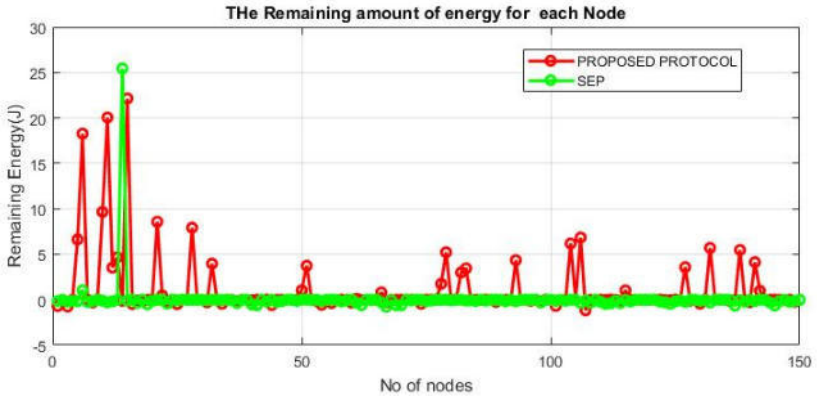


Fig. 5.7: The remaining amount of energy

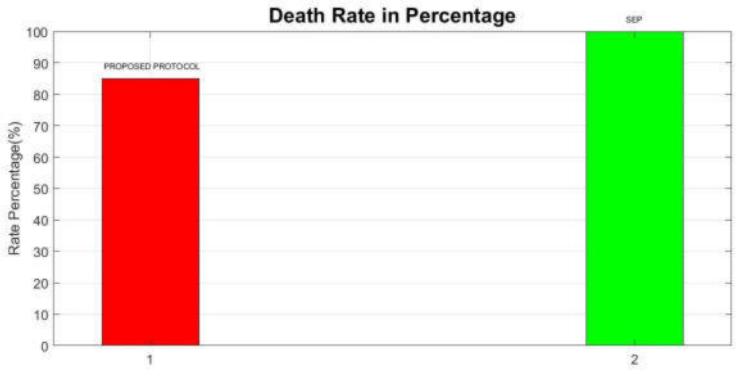


Fig. 5.8: The Death Rate of Network

SEP protocol. It is observed that the energy consumption is around 100 % in SEP protocol for 1400 rounds whereas it is around 80% for 1400 rounds in the proposed protocol.

In the Figure 5.7 the proposed protocol has more remaining energy compare to the SEP protocol which has almost zero energy for each node.

In the Figure 5.8 the survival rate of the proposed protocol is around 15% which is significantly better in comparison with SEP.

6. Conclusions. The proposed protocol prevents randomness in the selection process of CH. The new Threshold value consists of weighted energy and distance ratio which prevent the energy issues inside the sensor network. The data fusion method is used in the Proposed protocol to get accurate data using the trust function and the energy model is utilized to minimize energy trnasmission within the network. The simulation results have shown a better lifetime, stability period and survival rate for the Proposed protocol compared with the SEP protocol. In the future fuzzy logic approach will be used for the CH selection process which efficiently utilized energy consumption inside the network and enhanced the system performance.

REFERENCES

[1] Munir, Saad Ahmed and Ren, Biao and Jiao, Weiwei and Wang, Bin and Xie, Dongliang and Ma, Jian. Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing. *C*luster Computing, 2(3), pp. 113–120,(2007). Springer.

[2] V. Narayan, A. Daniel, Novel protocol for detection and optimization of overlapping coverage in wireless sensor networks (2019).

[3] Roy, Nihar Ranjan and Chandra, Pravin. Analysis of data aggregation techniques in wsn. *C*luster Computing, 22(3), pp. 571–581,(2020). Springer.

[4] Narayan, Vipul and Daniel, AK and Rai, Ashok Kumar. Energy Efficient Two Tier Cluster Based Protocol for Wireless Sensor Network. *C*luster Computing, 22(3), pp. 574–579,(2020). Springer.'

[5] Chaturvedi, Pooja and Daniel, AK. Trust based node scheduling protocol for target coverage in wireless sensor networks. *C*luster Computing, 22(3), pp. 163–173,(2015). Springer.

[6] Chaturvedi, Pooja and Daniel, AK. Trust based energy efficient coverage preserving protocol for wireless sensor networks. *C*luster Computing, 22(3), pp. 860–865,(2015). Springer.

[7] Chaturvedi, Pooja and Daniel, Ajai K. Trust Based Target Coverage Protocol for Wireless Sensor Networks Using Fuzzy Logic. *C*luster Computing, 22(3), pp. 188–192,(2016). Springer.

[8] Tripathi, Abhishek and Gupta, Hari Prabhat and Dutta, Tanima and Mishra, Rahul and Shukla, KK and Jit, Satyabrat. Coverage and connectivity in WSNs: A survey, research issues and challenges. *I*EEE Access, 6(3), pp. 26971–26992,(2018). IEEE.

[9] Rajpoot, Prince and Dwivedi, Pragya. Optimized and load balanced clustering for wireless sensor networks to increase the lifetime of WSN using MADM approaches. *W*ireless Networks, 26(1), pp. 215–251,(2020). Springer.

[10] Yadav, Ravi and Daniel, AK. Fuzzy based smart farming using wireless sensor network. *C*luster Computing, 22(3), pp. 1–6,(2018). Springer.

[11] Lu, Yu-ding and Chen, Yao-dong and Chen, Meng-yuan. The improvement and simulation research of wireless sensor network LEACHprotocol. *J*ournal of Anhui Polytechnic University, 22(4), pp. 13,(2012). Springer.
[12] Shokouhifar, Mohammad and Jalali, Ali. A new evolutionary based application specific routing protocol for clustered wireless sensor networks. *A*EU-International Journal of Electronics and Communications, 69(1), pp. 432–441,(2015). Elsevier.
[13] Zhenxing, Wang and Weili, Xiong and Baoguo, Xu. A LEACH Cluster Tree Network Routing Algorithm Research [J]. *C*omputer Measurement & Control, 11(3), pp. 5811–5823,(2008). Springer.
[14] Jiang, JianMing and Shi, GuoDong and Zhao, Dean and Li, ZhengMing and Shi, Bing and Zhao, YiGang et al. Intelligent monitoring system of aquaculture parameters based on LEACH protocol.. *N*ongye Jixie Xuebao= Transactions of the Chinese Society for Agricultural Machinery, 45(11), pp. 286–291,(2014). Chinese Society for Agricultural Machinery.
[15] Li, Fangfang and Wang, Jing. A New LEACH-Based Routing Algorithm for Wireless Sensor Networks [J]. *C*hinese Journal of Sensors and Actuators, 10(3), pp. 5811–5823,(2012). Springer.
[16] Wan, Chuanfei and Du, Shangfeng. Improvement and simulation of leach in wireless sensor networks. *J*isuanji Yingyong yu Ruanjian, 28(4), pp. 113–116,(2011). Shanghai Institute of Computing Technology.
[17] Roshan, Komal and Sharma, Kritika Rai. Improved LEACH protocol with cache nodes to increase lifetime of wireless sensor networks. *C*luster Computing, 22(3), pp. 903–908,(2018). Springer.
[18] Zhang, Li. The improvement and simulation of LEACH clustering routing protocol for WSNs. *W*uhan University of Technology, Wuhan, 22(3), pp. 1–75,(2009). Springer.
[19] Zayoud, Maha and Abdulsalam, Hanady M and Al-Yatama, A and Kadry, Seifedine. Split and merge leach based Routing algorithm for wireless sensor networks. *I*nternational Journal of Communication Networks and Information Security, 10(1), pp. 155–162,(2018). Springer.
[20] Xu, Yan and Yue, Zhanwei and Lv, Lingling. Clustering routing algorithm and simulation of internet of things perception layer based on energy balance. *I*EEE Access, 7(3), pp. 145667–145676,(2019). IEEE.
[21] Gawade, Rohit D and Nalbalwar, Sanjay L. A centralized energy efficient distance based routing protocol for wireless sensor networks. *J*ournal of Sensors, 2016(3), pp. 5811–5823,(2016). Hindawi.
[22] Wu, Wenliang and Xiong, Naixue and Wu, Chunxue. Improved clustering algorithm based on energy consumption in wireless sensor networks. *I*et Networks, 6(3), pp. 47–53,(2017). IET.
[23] Dhand, Geetika and Tyagi, SS. SMEER: Secure multi-tier energy efficient routing protocol for hierarchical wireless sensor networks. *W*ireless Personal Communications, 105(1), pp. 17–35,(2019). Springer.'
[24] Dwivedi, AK and Sharma, AK and Kumar, R. Dynamic Trust Management Model for the Internet of Things and Smart Sensors: The Challenges and Applications. *R*ecent Patents Comput. Sci, 12(3), pp. 5811–5823,(2019). Springer.
[25] Sun, Guiling and Zhang, Ziyang and Zheng, Bowen and Li, Yangyang. Multi-Sensor Data Fusion Algorithm Based on Trust Degree and Improved Genetics. *S*ensors, 19(9), pp. 2139,(2019). Multidisciplinary Digital Publishing Institute.
[26] Kolomvatsos, Kostas and Anagnostopoulos, Christos and Hadjiefthymiades, Stathes. Data fusion and type-2 fuzzy inference in contextual data stream monitoring. *I*EEE Transactions on Systems, Man, and Cybernetics: Systems, 47(8), pp. 1839–1853,(2016). IEEE.
[27] Karthick, Suyambu. TDP: A novel secure and energy aware routing protocol for wireless sensor networks. *I*nternational Journal of Intelligent Engineering and Systems, 11(2), pp. 76–84,(2018). Springer.
[28] Sharma, Richa and Vashisht, Vasudha and Singh, Umang. eeTMFO/GA: a secure and energy efficient cluster head selection in wireless sensor networks. *T*elecommunication Systems, 22(3), pp. 1–16,(2020). Springer.
[29] Mishra, Mukesh and Gupta, Gourab Sen and Gui, Xiang. Trust-Based Cluster Head Selection Using the K-Means Algorithm for Wireless Sensor Networks. *C*luster Computing, 22(3), pp. 819–825,(2019). Springer.
[30] Al-Humidi, Nada and Chowdhary, Girish V. Energy-aware approach for routing protocol by using centralized control clustering algorithm in wireless sensor networks. *C*luster Computing, 22(3), pp. 261–274,(2019). Springer.
[31] Gopalakrishna, Aravind Kota and Pai, Manohara MM. Multi-service adaptable routing protocol for wireless sensor networks. *C*luster Computing, 22(3), pp. 5811–5823,(oct " 23" 2012). Google Patents.
[32] Heinzelman, Wendi Beth. Application-specific protocol architectures for wireless networks. *C*luster Computing, 22(3), pp. 5811–5823,(2000). Springer.
[33] Smaragdakis, Georgios and Matta, Ibrahim and Bestavros, Azer. SEP: A stable election protocol for clustered heterogeneous wireless sensor networks. *C*luster Computing, 22(3), pp. 5811–5823,(2004). Springer.
[34] Dwivedi, Anshu Kumar and Sharma, Awadhesh Kumar and Mehra, Pawan Singh. Energy Efficient Sensor Node Deployment Scheme for Two Stage Routing Protocol of Wireless Sensor Networks assisted IoT. *E*CTI Transactions on Electrical Engineering, Electronics, and Communications, 18(2), pp. 158–169,(2020). IEEE.'

Edited by: Dana Petcu
Received: Sep 30, 2020
Accepted: Jan 3, 2021


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 13–28, DOI 10.12694:/scpe.v22i1.1818

MITIGATING MALICIOUS INSIDER ATTACKS IN THE INTERNET OF THINGS USING SUPERVISED MACHINE LEARNING TECHNIQUES

MIR SHAHNAWAZ AHMAD* AND SHAHID MEHRAJ SHAH†

Abstract. The interconnection of large number of smart devices and sensors for critical information gathering and analysis over the internet has given rise to the Internet of Things (IoT) network. In recent times, IoT has emerged as a prime field for solving diverse real-life problems by providing a smart and affordable solutions. The IoT network has various constraints like: limited computational capacity of sensors, heterogeneity of devices, limited energy resource and bandwidth etc. These constraints restrict the use of high-end security mechanisms, thus making these type of networks more vulnerable to various security attacks including malicious insider attacks. Also, it is very difficult to detect such malicious insiders in the network due to their unpredictable behaviour and the ubiquitous nature of IoT network makes the task more difficult. To solve such problems machine learning techniques can be used as they have the ability to learn the behaviour of the system and predict the particular anomaly in the system. So, in this paper we have discussed various security requirements and challenges in the IoT network. We have also applied various supervised machine learning techniques on available IoT dataset to deduce which among them is best suited to detect the malicious insider attacks in the IoT network.

Key words: Internet of Things, attack detection, security, malicious insider, supervised machine learning.

AMS subject classifications. 68M14

1. Introduction.

The recent advancement in various technological fields has led to the interconnection of enormous devices over the Internet, thus giving rise to the Internet of Things (IoT) network. With the help of IoT network even the ordinary devices (e.g. wearables, smart meters, smart water meters etc.) used by human beings in day-to-day living can be used to gather the information from surroundings using sensors/actuators, which can be used to solve various real life problems. IoT has transformed the classical field into smart field like: smart healthcare, smart transport, smart grid, smart home, smart waste management and many more [1]. However, an IoT network has some constraints like, limited computational capability of sensors, heterogeneity of devices, limited energy resource and bandwidth etc. These constraints restrict the use of high-end security mechanisms in IoT network and thereby makes such networks more vulnerable to malicious insider attacks[2].

A malicious insider can be employee or an ex-employee or a business partner of a company, who has or once had an authorized access to the company's data or network. Malicious insider can exploit that access to launch various malicious attacks in the company's network [3]. Non-authorized attackers are comparatively easy to detect since they have to break various authentication and authorization mechanisms employed by a company. Whereas, an insider is already known to company's security mechanisms and gets an easy access to the company's network and crucial data, thus making it hard for the already implemented security mechanisms to detect such attackers in the network. With the increasing use of IoT devices the insider attacks may also increase drastically because we are surrounded by a large number of IoT devices. Since IoT devices are usually low-end devices with limited resources hence it becomes easy for the insider attackers to launch malicious attacks in the network.

To solve such problem of detecting insider attackers in an IoT network, machine learning techniques can be used as they have the ability to learn the behaviour of the system and predict a particular anomaly in the system[4]. In order to use various machine learning techniques to detect malicious insiders in the IoT network,

*Communication Control & Learning Lab, Department of Electronics & Communication Engineering, National Institute of Technology, Srinagar, J&K, India. (mirshahnawaz888@gmail.com).

†Communication Control & Learning Lab, Department of Electronics & Communication Engineering, National Institute of Technology, Srinagar, J&K, India.

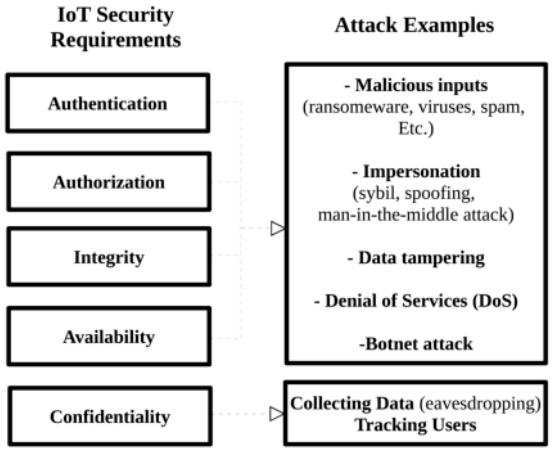


Fig. 2.1: Attacks that can affect the security requirements of IoT devices.

we need some dataset to train the algorithm. One such dataset is Network Security Laboratory–Knowledge Discovery in Databases (NSL-KDD). NSL-KDD is the latest version of KDD99 dataset [5]. The KDD99 was generated in 1999 as a result of international competition for Knowledge Discovery in Databases (KDD) and using DARPA98 network traffic. The various attacks included in NSL-KDD dataset include: DoS (Denial of Services) attack, User-to-Root attack, Remote-to-local attack and Probes. In the literature, this dataset has been widely used to assess the performance of anomaly-based attack detection systems for IoT network [6, 7, 8].

In this paper we apply various supervised machine learning algorithms on NSL-KDD dataset and analyse the performance of each algorithm. Based on various performance metrics we identify the best machine learning technique for malicious insider attack detection.

The rest of the paper is organized as: Section 2 describes various IoT security requirements and challenges; Section 3 highlights the characteristics of a malicious insider, its types and various malicious activities carried out by such attacker in an IoT network; Section 4 describes various supervised machine learning techniques used in our study; Section 5 outlines the attributes of NSL-KDD dataset and describes various performance parameters to measure the performance of each machine learning classifier for detecting a malicious insider; Finally, section 6 concludes the paper.

2. IoT security requirements and challenges. IoT is the integration of physical objects/things over an internet in order to create a smart living environment. The IoT devices are usually deployed in diverse environment to sense or gather the desired data, which can be used to accomplish various tasks. However, to successfully accomplish a task using IoT network, it must meet various security requirements [9]. The heterogeneous nature of IoT network makes it susceptible for various attacks. Due to the limited computational resources of IoT devices, it becomes computationally difficult to use complex security mechanisms for attack detection. The IoT devices are usually connected over a wireless medium, this makes them vulnerable to malicious intruders. Figure 2.1 shows some of the attacks that can affect various security requirements of IoT network, which are summarized as:

- *Authentication:* The identity of IoT network user should be first verified and only then they should be allows to access the network. However, due to limited resources of IoT devices the implementation of robust authentication mechanism is a challenging task. The authentication mechanism should have a trade-off between robustness, flexibility and IoT device constraints [10].
- *Authorisation:* It includes those security procedures which grant permission to legitimate users so that they can access various IoT devices in a network [2].
- *Integrity:* The sensed data by IoT devices is usually transferred/ shared via wireless channels, due to which it can be easily accessed by illegitimate users that can then modify it. So, one must ensure

integrity of IoT network data and should be flexible enough to deal with wide variety of IoT devices [2].

- *Availability:* The services provided by various IoT devices should be readily available to all the authorized users, but attacks like Denial of Services (DoS) and jamming attacks may create a hindrance by overwhelming the IoT devices. So, we need to implement such mechanisms which can detect such attacks in IoT network, but these attacks are difficult to detect and the heterogeneity & resource constraints of IoT network make it much more challenging task.
- *Confidentiality:* Like other networks, the security of gathered data in IoT network is very important. The confidentiality of data can be done by implementing various encryption algorithms, but keeping in view the resource constraints of IoT devices, these algorithms should be lightweight [11].

In this paper we deal with attacks related to availability requirement for the IoT network.

3. Malicious Insider Attacks in IoT. In this section we will be discussing in detail about the malicious insider attacks, their types and how they can affect the IoT networks.

3.1. Malicious Insider. Greitzer et al. [12] defines a malicious insider as a user of an organization who once had or has currently an authorized access to the organization's confidential data, resources or network, and uses these authorized privileges to launch various attacks in the network. Since the malicious insider has an authorized access to the network, so the authentication/ authorization mechanism implemented at the organizational level is unable to detect such attackers in the network. Due to this characteristic of an insider attacker node, it becomes very challenging for a security mechanism to detect such attackers in the network. Also, if these attackers are left undetected in an IoT network, it makes IoT nodes vulnerable to a malicious node. In order to further understand the details of insider attackers, we will be discussing various types of malicious insiders in the following section.

3.2. Types of Malicious Insiders. The malicious insiders can be categorized mainly into three categories: traitor, masquerader and unintended insiders. Among all these insiders, traitor is most threatening to a system and is responsible for launching around 92% of total attacks out of all the malicious insider attacks [13]. The traitors are those attackers who have authorized access to company's resources and use these privileges to launch various attacks in the company's network. They can be more severe since they already know the vulnerabilities of a network and are independent of any time constraint for launching attacks. Another category of insider attackers is a masquerader [14] who does not belong to an organization but somehow gets an access to the organization's networks (e.g. by guessing/ stealing the password of a legitimate user). The unintended insiders can also be categorized as a malicious insider, who threatens an organization by accidentally breaking the security mechanism [15].

3.3. Classification of Malicious Insider activities. Various activities that a malicious insider can perform in order to harm an organization mainly includes: IT Sabotage, Theft of Intellectual Property, Fraud and Espionage [9]. In IT sabotage, a malicious insider makes use of privileged access to network data and uses it to directly vandalise an organization or an individual by lunching attacks like DoS, sybil, man-in-the-middle, botnet and various other attacks. Since a malicious insider can access the organization's resources, so it can steal various innovative ideas, schemes and other essential artefacts which constitute various intellectual properties of an organization. A malicious insider can also perform fraudulent transactions that can affect an organization. The espionage includes all those activities with which a malicious insider can overhear the network data and sell the organization's critical information to other rival companies.

All the above discussed malicious activities by an insider can critically harm an organization and when an organization uses IoT network, then these attacks become more catastrophic due to low computational abilities of IoT devices. So, one needs to use a security procedure that will not only detect such anomalies in the system but also puts less load on IoT devices. The characteristics of machine learning techniques make them suitable for detecting such malicious attacks in the IoT network.

4. Supervised machine learning algorithms for detecting malicious insiders in IoT network. In this section we will discuss various supervised machine learning techniques that are used in our study.

4.1. Ridge Regression. If the data is given as (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, then a linear regression model can be written as:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} \tag{4.1}$$

where, $\beta_1, \beta_2, \cdots, \beta_p$ represents the regression coefficients for different x_i. The aim of linear regression is to choose those β_i for which the residual sum of squares minimize. Hence the optimization problem is framed as:

$$\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] \tag{4.2}$$

But, this model does not generalize well for the new data (i.e., the data with high variance) and hence result into overfitting. To overcome this problem, the ridge regression is used, which continuously narrows down the regression coefficients and hence producing a stable model [16]. The ridge regression overcomes this problem by adding a constraint for optimization problem of 4.2. Hence the optimization problem corresponding to a ridge regression can be written as:

$$\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right], s.t. \sum_j |\beta_j|^2 \leq 1 \tag{4.3}$$

Now to solve this optimization problem, we introduce Lagrange multiplier α, also known as regularization constant. Hence, Ridge estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \alpha \sum_j |\beta_j|^2 \tag{4.4}$$

subjected to the condition that for each x_{ij}, $\sum_i x_{ij}/N = 0$ and $\sum_i {x_{ij}}^2/N = 1$. Where, $\sum_j |\beta_j|^2$ is the regularization term and α is constant which controls the amount of regularization applied. In our study, we have set $\alpha = 1$ and allowed maximum iterations to 10.

4.2. Lasso Regression. In a model where a set of features are highly correlated the ridge regression will distribute weights equally to all the correlated features. Also, ridge regression will shrink the coefficients of less important predictors, but never makes them zero. This hinders the process of feature selection in a model. To overcome these limitation Lasso (Least Absolute Shrinkage and Selection Operator) regression can be used, which narrows down some of the model coefficients and sets others to zero [17]. It combines the best features of ridge regression and subset selection. For a given data (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, the Lasso estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \alpha \sum_j |\beta_j| \tag{4.5}$$

subjected to the condition that for each x_{ij}, $\sum_i x_{ij}/N = 0$ and $\sum_i {x_{ij}}^2/N = 1$. where α is the constant which controls the amount of regularization applied. The penalty term $\sum_j |\beta_j|$ has such effect that it forces some of the coefficients to exactly zero for large value of α. Thus, helping in variable selection and yielding sparse models. In our study, we have set $\alpha = 0.019$ and allowed maximum iterations to 150.

4.3. Elastic Net. The Lasso only selects one variable among a group of variables having high correlation and due to convex optimization it selects a limited number of features. These properties of Lasso affects its performance. To overcome this issue, elastic net regularization technique was proposed [18]. The elastic net can continuously shrink the model parameters and can also select a group of variables which have high correlation.

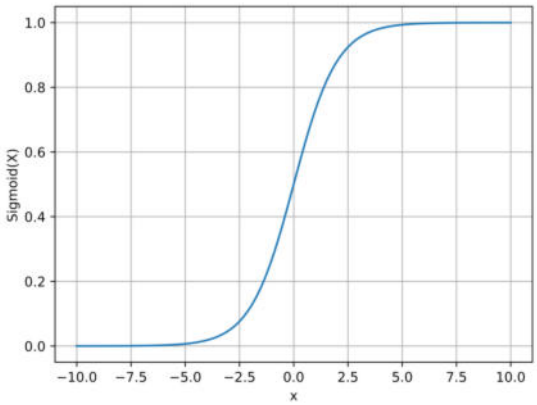


Fig. 4.1: Sigmoid Function

If the data is given as (x_i, y_i) where, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T$ is the input and y_i is the outcome, then the elastic net estimate for $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_p)^T$ is given as:

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{N} (y_i - \sum_j \beta_j x_{ij})^2 \right] + \lambda_2 \sum_j |\beta_j|^2 + \lambda_1 \sum_j |\beta_j| \tag{4.6}$$

under the condition that: $\sum_{i=1}^{n} y_i = 0$, $\sum_{i=1}^{n} x_{ij} = 0$, and $\sum_{i=1}^{n} {x_{ij}}^2 = 1$, for $j = 1, 2, \cdots, p$. where λ_1 and λ_2 are non-negative constants. In our study, we have set $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$ with allowed maximum iteration to 150.

4.4. Lasso with Lars. Lars is a least-angle regression technique which helps to fit regression model to a high dimensional data. In our study we have used the combination of Lasso with Lars which resembles forward step-wise regression process, but at each step it selects those estimated coefficients which are in the direction equiangular with the square of residual error [19]. In its implementation for detecting malicious traffic in the IoT dataset, we have set α (constant which controls the amount of regularization applied) = 0.02 and maximum iteration to 50.

4.5. Logistic Regression (LR). Logistic regression describes a statistical method of predicting a binomial event. Like other Machine Learning classifiers, the input data can consist of one or multiple features. The outcome for the binary logistic regression is 0 or 1 that performs a differential positive class classification from the negative class. To give out a probabilistic value, Sigmoid curve shown in figure 4.1 is used in logistic regression [20].

The hypothesis function used in logistic regression in shown in equation below:

$$h(x) = S(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) \tag{4.7}$$

where $x_1, x_2, \ldots, x_n$ denotes the features, $w_0, w_1, w_2, \ldots, w_n$ denotes the model weights and $S()$ represents the sigmoid function identified by:

$$S(Z) = \frac{1}{1 + e^{-Z}} \tag{4.8}$$

The range of output of $S()$ is from 0 to 1. All the values below 0.5 refer to negative class while as values from 0.5 to 1 indicate a positive class. While implementing logistic regression on the dataset we have set maximum iteration of 100 and used $\mathcal{L}^2$ norm for regularization, also we have used BFGS (Broyden Fletcher Goldfarb Shanno) optimization algorithm [21].

4.6. K-Nearest Neighbors (KNN). In KNN algorithm, the aim is to classify any new, unknown data point by analyzing the data points (most similar to the input or feature space) given in the training set [22]. The algorithm works by finding the K-nearest neighbours of the new data point by usually using one of the three distance measures for continuous variable viz. Euclidean distance (EuD), Manhattan distance (MaD) or the Minkowski distance (MiD) represented below:

$$EuD = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.9}$$

$$MaD = \sum_{i=1}^{n}|x_i - y_i| \tag{4.10}$$

$$MiD = \left[\sum_{i=1}^{n}(x_i - y_i)^q\right]^{1/q} \tag{4.11}$$

where, $x = (x_1, x_2, \ldots, x_n)^T$ & $y = (y_1, y_2, \ldots, y_n)^T$ represent the data points.

We have implemented the KNN classifier for various values of K on the given dataset. To choose the optimal value of K we observed that the accuracy of KNN algorithm increases with K but for $K > 10$ the accuracy saturates, hence choose the value of K as 10. We have used Euclidean distance to find K-nearest neighbours for a new data point.

4.7. Support Vector Machine (SVM). It is a Machine Learning classifier that is currently receiving the most significant attention due to its high performance [23] and the ability to alleviate the problem of classifying non-linear data. Typically SVMs are the non-probabilistic, binary classifiers aimed at discovering a hyperplane that divides the two classes of the training set with the highest margin.

If the data is given as $(\vec{x_i}, y_i)$ where, $\vec{x_i}$ represent the input vector and y_i is the outcome which indicates to which a particular $\vec{x_i}$ belongs. The SVM tries to find a maximum-margin hyperplane which divides $\vec{x_i}$ into different classes such that the distance between the hyperplane and the nearest point $\vec{x_i}$ from either classes is maximized. The equation of a hyperplane is given as:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{4.12}$$

where, $\vec{x}$ represent the input vector, $\vec{w}$ is the vector normal to hyperplane and b is the intercept (bias term). The objective of SVM is to minimize:

$$\left[\frac{1}{p}\sum_{i=1}^{p}\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))\right] + \alpha\|\vec{w}\|^2 \tag{4.13}$$

where, $\max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b))$ represent the hinge loss function, which return a value proportional to distance from margin to $\vec{x_i}$, if $\vec{x_i}$ is misclassified, otherwise returns zero. $\|\vec{w}\|^2$ represents the penalty for incorrect classification, which is controlled by constant α.

Apart from linear classification, SVMs perform non-linear classification of data by implicitly mapping an input variable into high dimensional attribute spaces using kernel trick [24].

As such SVMs tried in our work are classified into three categories: Linear SVM, SVM with Radial Basis Function (RBF) kernel and SVM with the polynomial kernel. The information about their kernels is given below.

Radial Basis Function (RBF) is the common kernel choice that is implemented in SVM. Given x and y as the input feature vectors, then the RBF kernel is given as:

$$K(x, y) = \exp(-\gamma||x - y||^2) \tag{4.14}$$

where, γ defines the influence of single training example in a model, i.e., increased value of γ results into over-fitting and decreased value results into under-fitting of model. $\gamma > 0$, and is often parametrized by $\gamma = \frac{1}{2\sigma^2}$, where σ is a free independent parameter. Therefore, the RBF kernel becomes:

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \tag{4.15}$$

To define the similarity among the input samples, Polynomial kernel not only takes into account the given features, but also the combinations of samples. A polynomial kernel is given as:

$$K(x, y) = (x^T \cdot y + t)^c \tag{4.16}$$

where x, y denote input feature vectors, t is a complexity parameter which trades-off between higher-order and lower-order terms in the polynomial ($t \geq 0$), and c is the integer exponent. If $t = 0$, then the kernel is homogeneous.

In our study, for SVM, we have used $\mathcal{L}^2$ norm for regularization in all kernel types. For Linear SVM, maximum iteration was set to 10000 and kernel coefficient was $1/N$, where N is the total features in dataset. For polynomial kernel type the degree was chosen to be 3.

4.8. LR and SVM with Stochastic Gradient decent (SGD). SGD is an extended version of Gradient Descent algorithm that decreases the number of computation per iteration by incorporating randomness in the learning process. It repeatedly considers a single or batch training example and evaluates the gradient of the loss function to reduce the model's risk, and hence updates the parameters used by the model. Whether it be a strictly convex problem or a non-convex one, SGD performs better and supports robustness and scalability [25]. The main objective of SGD algorithm is to minimize the regularized training error, which is the combination of empirical risk and the regularization term. The empirical risk term measures the performance of training set using a loss function $L(\hat{y}, y)$ (that gives the cost of predicting $\hat{y}$, given y as the actual output). The regularization term decides the penalty for the model's complexity, whose impact is decided by the term α (non- negative hyperparameter) [26].

We have studied the performance of SGD using SVM and logistic regression. SVM uses Max-margin classification, where correct category score should be larger (by a predefined margin) than the collective wrong category scores. Log loss (or cross-entropy loss or the logistic loss) decreases as the probability of correct prediction increases. During simulation, maximum number of iterations was set to 10000 with $\alpha = 0.1$, where α is a constant which is multiplied to regularization term to control the degree of regularization. Here again we have used $\mathcal{L}^2$ for regularization.

4.9. Random Forest (RF) Classifier. Random Forest Classifier is an Ensemble method of classification, which combines the outputs of various decision trees to generate more accurate results. Thus, it is based on the simple philosophy that the collection of classifiers will always perform better than a single classifier. This classifier works well for large datasets, gives an estimate of essential features in the dataset and possesses robustness for noise and outliers [27]. It also decreases the generalization error by randomly selecting subset of features during the splitting of a node (which also decreases the degree of correlation between various generated trees). Random Forest uses the Gini Index (GI) as an approximation to choose the best set of features during tree formation.

$$GI = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{4.17}$$

where, p_i is the probability with which an element is classified to a particular class. Due to the use of multiple trees, some of the features in the dataset may be used more than once for the training purpose. This increases the classifier stability because a slight change in input may not cause any change in the output of classifier, hence making it more robust and accurate [27].

In our study we have chosen the best split at each node in a decision tree using gini index with a minimum allowed splits at each node as 2. The classifier was iteratively run on the training data to find the optimal

number of trees in the forest and maximum training samples to be used to train each base estimator. We observed that if we increase the number of trees in the forest beyond 10 and maximum training samples to be used for base estimator training beyond 1000, then there was no change in classifier's accuracy. So, we selected number of trees in the forest as 10 and maximum training samples to be used for base estimator training as 1000.

4.10. AdaBoost (AB). AdaBoost is an iterative machine learning technique which attempts to fit weak classifiers iteratively [28]. The process starts with un-weighted training sample, using which a weak classifier is trained and then at each iteration the weights are boosted and the classifier is trained again. At each iteration the weights of those examples are increased for which the classifier at previous iteration predicted wrong label and the weights for those examples are decreased for which the classifier at previous iteration classified correctly. This process continues until all the data entries are correctly classified or it becomes difficult for a classifier to predict a label for unclassified/misclassified data. In this way the final classifier acts as the linear combination of various classifiers used at each iteration.

In our study, we have used Decision tree classifier as the weak classifier at each step and the maximum size of estimator (at which it terminates) was set to 200. The learning rate was set to 1 and we have used SAMME (Stage-wise Additive Modelling using a Multi-class Exponential loss function) algorithm for implementing the boosting process [29].

4.11. Gradient Boosting (GB). Gradient boosting is also a form of ensemble method which uses gradient descent to minimize the model's loss function by incorporating weak classifiers in an iterative manner [30]. At each iteration the gradient descent algorithm selects those classifiers which minimize the loss function. In this technique each weak classifier is trained on the residue of strong classifier. The input to the algorithm is the training data, a loss function (which should be differentiable) and number of iterations. In each iteration, it computes the residual error from strong classifiers, uses it to fit the weak classifiers, chooses that classifier which minimizes the loss function and finally updates the model.

We have used Decision tree classifier as the weak classifier at each step and log loss function with learning rate of 0.1 in our study. Number of boosting stages was chosen to be 200 and the quality of node split in decision tree was done using mean squared error (with an improved score by Friedman).

4.12. Artificial Neural Network (ANN) Classifier. For ANN based classification we have used multi-layer perceptron (MLP) classifier. MLP is a multi-layer feed-forward neural network that trains using back-propagation algorithm [31]. The first layer of neurons acts as the input layer, which is responsible for taking inputs (feature vector), and the final layer is responsible for generating the classified output. In between these two layers, there can be multiple layers of neurons, known as hidden layer, which processes the input vector and helps the output layer to generate output. In hidden layers each connection from one layer to another is associated with a weight vector. Figure 4.2 shows an example of Multi-layer perceptron with only one hidden layer of neurons and a single neuron at the output layer.

Each neuron is associated with an activation function, which generates an output depending on the values of previous layer neurons and the associated weights. The training algorithm (SGD) tries to fit a perfect combination of weight vector values to reduce the loss.

For implementing MLP, we have used a single hidden layer of neurons (with number of neuron = 90 and the activation function as a rectified linear unit function) and a single neuron at the output layer to classify the input data into malicious (abnormal) or non-malicious (normal). We train the neural network using SGD algorithm with a learning rate = 0.001 and $\mathcal{L}^2$ norm for regularization.

5. Numerical results. To compare the efficiency and applicability of various Machine Learning algorithms for detecting malicious insider attacks in IoT, we have used a popular Intrusion detection dataset – NSL-KDD, the details of which are given in next sub-section.

5.1. Introduction to NSL-KDD dataset and data pre-processing. NSL-KDD is the latest version of KDD99 dataset and has been widely used by researchers to evaluate the performance of the attack detection system in IoT [6, 7, 8]. In NSL-KDD dataset, the training data is available in "KDDTrain+" file, consisting of 125973 data entries, out of which 67343 entries represent non-malicious and 58630 entries represent malicious.

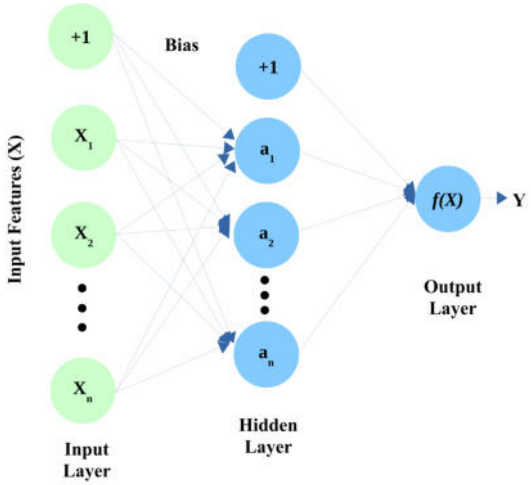


Fig. 4.2: Multi-layer Perceptron with single hidden layer and only one neuron in output layer

The testing data is available in "KDDTest+" file, consisting of 22543 entries, out of which 9710 represent non-malicious and 12833 entries represent malicious. Each dataset has 41 features and a single class label (as shown in figure 5.1). After analysing the feature of dataset, we have categorized them into four categories. Category – I represents those features which reveal the properties of TCP connection between source and destination node. Category – II highlights the basic features of the data that is being shared over a single connection between source and destination nodes. Category – III represents those features which depict the status of various connections in the network with a time frame of 2 seconds. Attributes of destination nodes in various connections in a network which help to analyse an attack that lasted for more than 2 seconds are represented by category – IV. The last feature represents the class label for each data entry (malicious or non-malicious) in the dataset. Among all the features, some of them represent various sub-features/attributes, they include: 'Protocol type', 'Service' and 'flag', and for the computational purposes we have converted them into numeric data. In 'protocol type' we have used 1, 2 and 3 for representing ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols respectively. The 'service' attribute contained 70 different web services used by the destination network. Various attributes of 'flag' feature are shown in figure 5.2.

Since the available data in the dataset is multidimensional and to enhance the performance of Machine Learning algorithm, we have normalized the data using standard scalar:

$$\hat{X}_i = \frac{X_i - \mu}{\sigma} \tag{5.1}$$

where, $\hat{X}_i$ is the normalized value of a data sample X_i with mean μ and standard deviation σ. It is evident from the above equation that the Standard Scaler normalizes the data by removing the mean of each feature and scaling the variance to 1.

5.2. Performance Parameters. To evaluate the performance of each trained machine learning model, confusion matrix was used, that is shown in Table 5.1.

Using the values obtained from the confusion matrix for each classifier, the following performance parameters were used to analyse the performance:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{5.2}$$

Category I	duration (0–to–54451)	protocol type (1, 2, 3)	service (1–to–70)	src_bytes (0–to–1379963888)	dst_bytes (0–to–309937401)
	Flag (1–to–11)	Land (0, 1)	wrong_fragment (0, 1, 3)	Urgent (0–to–3)	
Category II	Hot (0–to–101)	num_failed_logins (0–to–4)	logged_in (0,1)	num_compromised (0–to–7479)	root_shell (0, 1)
	su_attempted (0, 1)	num_root (0–to–7468)	num_file_creations (0–to–100)	num_shells (0–2)	num_access_files (0–9)
	is_guest_login (0, 1)	is_hot_logins (0, 1)	num_outbound_cmds (0)		
Category III	Count (0–511)	srv_count (0–511)	serror_rate (0–to–1)	srv_serror_rate (0–to–1)	rerror_rate (0–to–1)
	srv_rerror_rate (0–to–1)	same_srv_rate (0–to–1)	diff_srv_rate (0–to–1)	srv_diff_host_rate (0–to–1)	
Category IV	dst_host_count (0–to–255)	dst_host_srv_count (0–to–255)	dst_host_same_srv_rate (0–to–1)	dst_host_diff_srv_rate (0–to–1)	dst_host_same_src_port_rate (0–to–1)
	dst_host_srv_diff_host_rate (0–to–1)	dst_host_serror_rate (0–to–1)	dst_host_srv_serror_rate (0–to–1)	dst_host_rerror_rate (0–to–1)	dst_host_srv_rerror_rate (0–to–1)
	class_label (0–to–1)				

Fig. 5.1: Various features of NSL–KDD dataset with corresponding values in the dataset

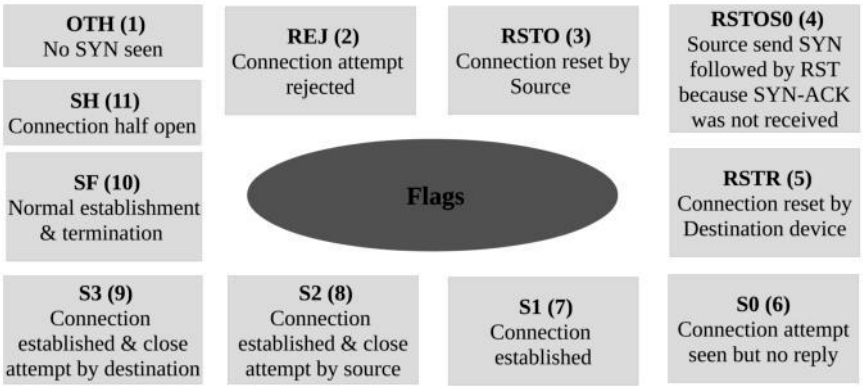


Fig. 5.2: Attributes of Flag feature in NSL–KDD Dataset

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

Table 5.1: Confusion Matrix

Actual Value	**Predicted Value**	
	Malicious	**Non-malicious**
Malicious	True Positive (TP)	False Negative (FN)
Non-malicious	False Positive (FP)	True Negative (TN)

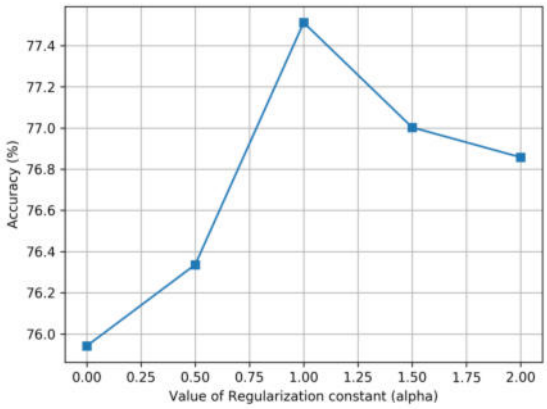


Fig. 5.3: Accuracy of Ridge regression for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of α

$$F1 - score = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.5}$$

The accuracy give the percentage of samples that are correctly classified out of all the tested samples, precision depicts the classifier's ability to detect only relevant data, recall gives the efficiency of a classifier to detect all interested data points in the dataset (i.e total positives detected by the system to that of actual positives throughout the system) and F1-score is the harmonic mean of both precision and recall.

We have also used Area Under the Curve (AUC) of a Receiver Operating Characteristic (ROC) curve as a performance matrix, which gives the overall performance of each classifier used in classifying the malicious and non-malicious traffic in KDDTest+ dataset. The value of AUC lies between [0, 1] and the performance of a particular classifier is directly proportional to the value of AUC obtained, i.e. a classification model is considered efficient if its AUC value lies close to 1 and inefficient if its value lies close to 0. Besides these performance parameters, we have also noted the time a particular classifier takes to train on KDDTrain+ dataset (Training Time) and the time it takes to predict a label for data in KDDTest+ dataset (Test Time). Evidently, more the training time of a particular classifier, more computation resources are needed for achieving classification, and lesser the test time indicates that the classifier will have quick response.

5.3. Performance evaluation of various machine learning algorithms for detecting insider attacks in IoT. In order to analyse the performance of various supervised machine learning algorithms for detecting insider attackers in IoT network, they were first trained on KDDTrain+ dataset and then their classification performance was evaluated using KDDTest+ dataset. For the implementation of these models, we have used intel core i3-5005U CPU @ 2.00GHz * 4, RAM: 4GB and operating system: ubuntu 14.04 LTS (64bit), using Scikit-learn and pandas package.

To choose the optimal value of various critical parameters in various classifiers, we have simulated the models for different values as shown in figure 5.3 to 5.8.

Figure 5.3 plots the accuracy of ridge regression to detect malicious IoT network traffic against various

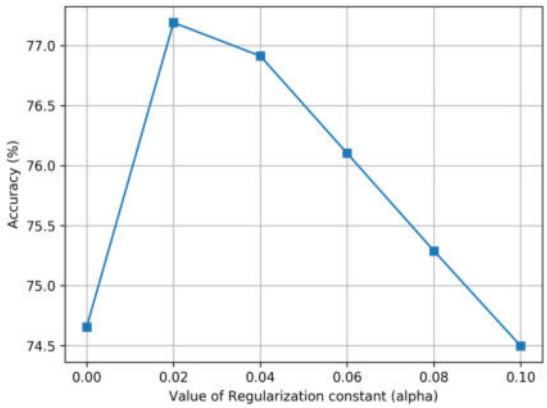


Fig. 5.4: Accuracy of Lasso regression for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of α

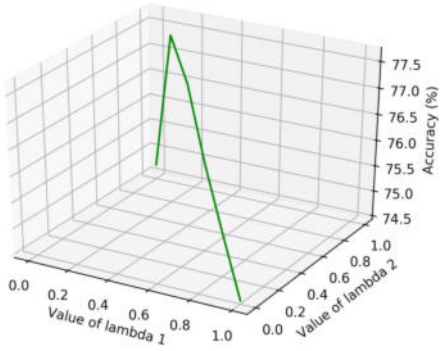


Fig. 5.5: Accuracy of Elastic Net for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of λ_1 and λ_2

values of α, and it is evident from the figure that maximum detection accuracy is achieved for $\alpha = 1$. Also, figure 5.4 proves that maximum detection accuracy is achieved at $\alpha = 0.02$ for Lasso regression. We have tested the Elastic Net for varying values of λ_1 and λ_2 on KDDTest+ dataset, as shown in figure 5.5. The figure shows that maximum accuracy is achieved for $\lambda_1 = 0.2$ and $\lambda_2 = 0.8$.

The value of K in K-Nearest Neighbor classifier was chosen to be 10, since the detection accuracy saturates after $K = 10$ (as shown in figure 5.6). Figure 5.7 gives the optimal number of iterations for various classifiers by analysing the detection accuracy. Optimal values for number of training samples used to train base estimator for Random forest, AdaBoost and Gradient Boosting respectively are shown in figure 5.8.

Thus, after getting the optimal values for various parameters, the detailed results for detecting malicious insider attacks in the IoT network by various supervised machine learning algorithms are shown in Table 5.2, figure 5.9 and 5.10.

After thorough analysis of performance parameters in table 5.2 and figure 5.9 & 5.10 for various machine learning algorithms, we conclude that ensemble machine learning methods (Random forest, AdaBoost and

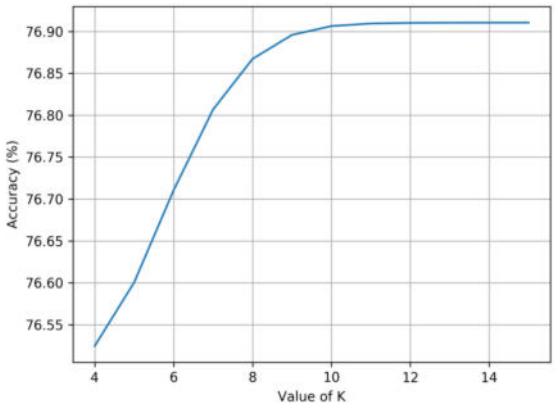


Fig. 5.6: Accuracy of K-Nearest Neighbor classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying values of K

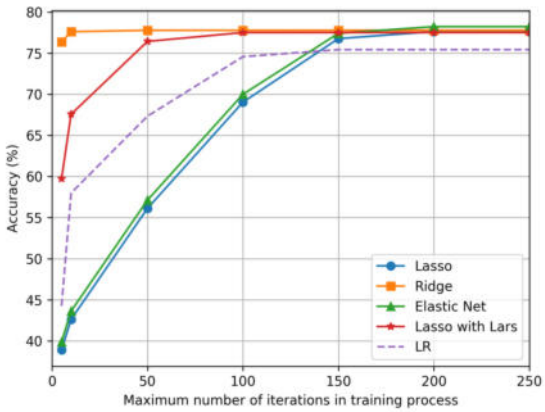


Fig. 5.7: Accuracy of various classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying number of iterations during training process

Gradient Boosting) perform better than the other studied machine learning classifiers for classifying the IoT network traffic into malicious and non-malicious. Among them the Gradient Boosting outperforms with an accuracy of 81.29% and AUC value of 0.96. Although the training time of Gradient Boosting algorithm is slightly more than some of the discussed algorithms, the testing time is nominal.

6. Conclusion. In this paper, we have discussed various characteristics and security requirements of an IoT network. We have also discussed how a malicious insider can be a major threat to IoT network and analysed various supervised machine learning algorithms which make them favourable for detecting such attacks in the IoT network. In order to demonstrate that the machine learning techniques can be used to detect malicious insiders in the IoT network, we trained various supervised machine learning algorithms on NSL-KDD dataset. The results show that Gradient Boosting technique outperforms the other discussed techniques. As part of future work, new and updated optimization techniques can be used to further enhance the attack detection accuracy of Gradient Boosting algorithm.

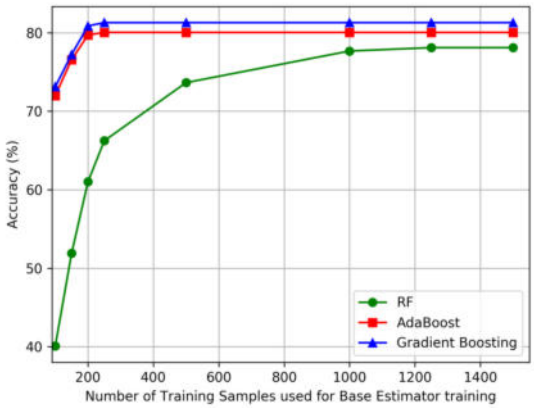


Fig. 5.8: Accuracy of various classifier for detecting malicious IoT network traffic in KDDTest+ dataset for varying number of training samples used to train base estimator

Table 5.2: Performance Analysis of various Machine Learning Classifiers on KDDTest+ dataset

Machine Learning Classifiers	**TP**	**FP**	**TN**	**FN**	**Accuracy (%)**	**Precision**	**Recall**	**F1-score**	**Training Time (Sec)**	**Test Time (Sec)**	**AUC**
Lasso	9391	4725	8108	319	77.62	0.83	0.78	0.77	0.93	0.0021	0.80
Ridge	9484	4786	8047	226	77.77	0.84	0.78	0.77	0.22	0.0021	0.80
Elastic Net	9404	4604	8229	306	78.22	0.84	0.78	0.78	0.62	0.0021	0.80
Lasso with Lars	9389	4752	8081	321	77.5	0.83	0.77	0.77	0.08	0.002	0.80
LR	9070	4899	7934	640	75.42	0.81	0.75	0.75	11.38	0.0021	0.87
K-NN	9482	4977	7856	228	76.91	0.84	0.77	0.77	74.39	184.52	0.85
Linear SVM	9071	4953	7880	639	75.19	0.81	0.75	0.75	410.21	0.0022	0.87
SVM with RBF Kernel	9508	4699	8134	202	78.25	0.84	0.78	0.78	117.93	9.68	0.94
SVM with Poly. Kernel	9500	5092	7741	210	76.48	0.83	0.76	0.76	100.81	7.85	0.89
LR with SGD	9342	4781	8052	368	77.16	0.83	0.77	0.77	0.50	0.0021	0.92
SVM with SGD	9515	4931	7902	195	77.26	0.84	0.77	0.77	0.36	0.0021	0.90
Random Forest	9430	4655	8178	280	78.10	0.84	0.78	0.78	14.05	0.32	0.96
AdaBoost	9409	4196	8637	301	80.05	0.85	0.80	0.80	34.18	0.79	0.95
Gradient Boosting	9424	3932	8901	286	81.29	0.86	0.81	0.81	73.47	0.11	0.96
MLP	9466	4581	8252	244	78.59	0.84	0.79	0.78	406.99	0.14	0.96

Acknowledgments. We would like to thank TEQIP-III and MITS, Gwalior for supporting this research.

REFERENCES

[1] F. Javed, M. K. Afzal, M. Sharif, and B.-S. Kim, *Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review*, in IEEE Communications Surveys & Tutorials, 20, No. 3 (2018), pp. 2062–2100.

[2] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, *Internet of things security: A survey*, in Journal of Network

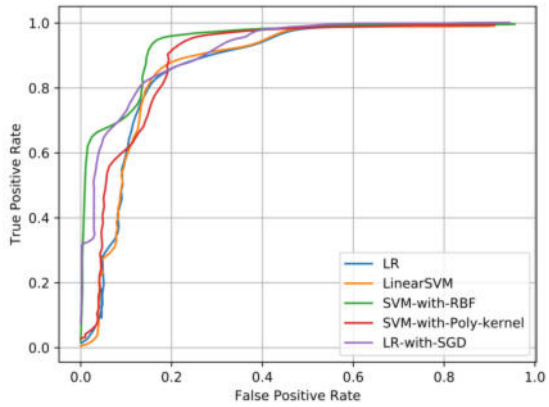


Fig. 5.9: ROC curves for various classifiers (having AUC > 0.85) after classifying IoT network traffic in KDDTest+ dataset (Part 1)

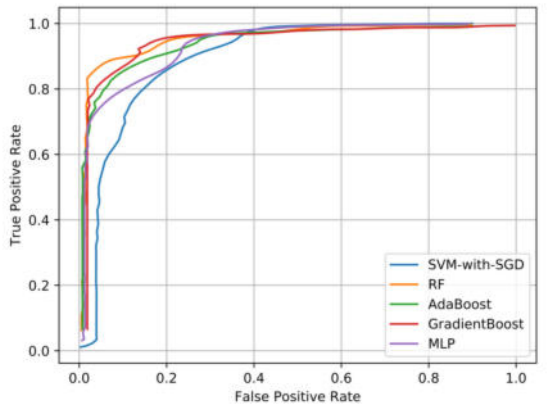


Fig. 5.10: ROC curves for various classifiers (having AUC > 0.85) after classifying IoT network traffic in KDDTest+ dataset (Part 2)

and Computer Applications, 88 (2017), pp. 10–28.

[3] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, *A survey on data leakage prevention systems*, in Journal of Network and Computer Applications, 62 (2016), pp. 137–152.

[4] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, *Iot security techniques based on machine learning: How do IoT devices use AI to enhance security?*, in IEEE Signal Processing Magazine, 35, no. 5 (2018), pp. 41–49.

[5] Nsl-kdd dataset, *https://www.unb.ca/cic/datasets/nsl.html*, (accessed: 02.08.2020).

[6] R. K. Gunupudi, M. Nimmala, N. Gugulothu, and S. R. Gali, *Clapp: A self constructing feature clustering approach for anomaly detection*, in Future Generation Computer Systems, 74 (2017), pp. 417–429.

[7] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, *Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset*, in IEEE Access, 8 (2020), pp. 29,575–29,585.

[8] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral, and G. dos Santos Vieira, *Hybrid approach to intrusion detection in fog-based IoT environments*, in Computer Networks, 180 (2020), pp. 107417.

[9] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, *Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures*, in ACM Computing Surveys (CSUR), 52, no. 2 (2019), pp. 1–40.

[10] J. Lopez, R. Roman, and C. Alcaraz, *Analysis of security threats, requirements, technologies and standards in wireless*

sensor networks, in Foundations of Security Analysis and Design V. Springer (2009), pp. 289–338.
[11] X. Yao, Z. Chen, and Y. Tian, *A lightweight attribute-based encryption scheme for the internet of things*, in Future Generation Computer Systems, 49 (2015), pp. 104–112.
[12] F. L. Greitzer and D. A. Frincke, *Combining traditional cyber security audit data with psychosocial data: towards predictive modeling for insider threat mitigation*, in Insider threats in cyber security. Springer (2010), pp. 85–113.
[13] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow et al., *Detecting insider threats in a real corporate database of computer usage activity*, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (2013), pp. 1393–1401.
[14] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, *Detecting and preventing cyber insider threats: A survey*, in IEEE Communications Surveys & Tutorials, vol. 20, no. 2 (2018), pp. 1397–1417.
[15] F. L. Greitzer, J. Strozer, S. Cohen, J. Bergey, J. Cowley, A. Moore, and D. Mundie, *Unintentional insider threat: contributing factors, observables, and mitigation strategies*, in 2014 47th Hawaii International Conference on System Sciences. IEEE (2014), pp. 2025–2034.
[16] D. W. Marquardt and R. D. Snee, *Ridge regression in practice*, in The American Statistician, 29, no. 1 (1975), pp. 3–20.
[17] R. Tibshirani, *Regression shrinkage and selection via the lasso*, in Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1 (1996), pp. 267–288.
[18] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, in Journal of the royal statistical society: series B (statistical methodology), vol. 67, no. 2 (2005), pp. 301–320.
[19] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani et al., *Least angle regression*, in The Annals of statistics, vol. 32, no. 2 (2004), pp. 407–499.
[20] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, in John Wiley & Sons (2013), vol. 398.
[21] R. Battiti and F. Masulli, *Bfgs optimization for faster and automated supervised learning*, in International neural network conference. Springer (1990), pp. 757–760.
[22] T. Cover and P. Hart, *Nearest neighbor pattern classification*, in IEEE transactions on information theory, vol. 13, no. 1 (1967), pp. 21–27.
[23] C. Cortes and V. Vapnik, *Support-vector networks*, in Machine learning, vol. 20, no. 3 (1995), pp. 273–297.
[24] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, *Machine learning for internet of things data analysis: A survey*, in Digital Communications and Networks, vol. 4, no. 3 (2018), pp. 161–175.
[25] G. Xu, Z. Cao, B.-G. Hu, and J. C. Principe, *Robust support vector machines based on the rescaled hinge loss function*, in Pattern Recognition, 63 (2017), pp. 139–148.
[26] S. Mehrkanoon, X. Huang, and J. A. Suykens, *Non-parallel support vector classifiers with different loss functions*, in Neurocomputing, 143 (2014), pp. 294–301.
[27] T. G. Dietterich, *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization*, in Machine learning, 40, no. 2 (2000), pp. 139–157.
[28] T. Hastie, S. Rosset, J. Zhu, and H. Zou, *Multi-class adaboost*, in Statistics and its Interface, vol. 2, no. 3 (2009), pp. 349–360.
[29] X. Zhu, P. Zhang, X. Lin, and Y. Shi, *Active learning from stream data using optimal weight classifier ensemble*, in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 40, no. 6 (2010), pp. 1607–1621.
[30] J. H. Friedman, *Stochastic gradient boosting*, in Computational statistics & data analysis, vol. 38, no. 4 (2002), pp. 367–378.
[31] A. Guezzaz, A. Asimi, A. Mourade, Z. Tbatou, and Y. Asimi, *A multilayer perceptron classifier for monitoring network traffic*, in International Conference on Big Data and Networks Technologies. Springer (2019), pp. 262–270.

Edited by: Dana Petcu
Received: Oct 15, 2020
Accepted: Jan 20, 2021


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 29–38, DOI 10.12694:/scpe.v22i1.1834

ESTIMATION OF TRAFFIC MATRIX FROM LINKS LOAD USING GENETIC ALGORITHM

JOSEPH L PACHUAU, ARNAB ROY, GOPAL KRISHNA AND ANISH KUMAR SAHA *

Abstract. Traffic Matrix (TM) is a representation of all traffic flows in a network. It is helpful for traffic engineering and network management. It contains the traffic measurement for all parts of a network and thus for larger network it is difficult to measure precisely. Link load are easily obtainable but they fail to provide a complete TM representation. Also link load and TM relationship forms an under-determined system with infinite set of solutions. One of the well known traffic models Gravity model provides a rough estimation of the TM. We have proposed a Genetic algorithm (GA) based optimization method to further the solutions of the Gravity model. The Gravity model is applied as an initial solution and then GA model is applied taking the link load-TM relationship as a objective function. Results shows improvement over Gravity model.

Key words: Traffic Matrix, Traffic Engineering, Gravity model, Link loads, Optimization, Genetic Algorithm, IP networks.

AMS subject classifications. 68M14

1. Introduction. Traffic matrix is a representation of traffic volume flowing between node pair in a network. It plays an important role in traffic engineering. Traffic matrix helps network manager in task like load balancing, network optimization, anomaly detection etc. The ingress-egress traffic matrix is collected between ingress and egress routers in the network. Origin destination matrix measures the traffic flow from actual source destination. The point where the packages are created and where they are received. For large IP networks this produces an extremely large and sparse matrix. For such a case an aggregated IP or blocks of IP may be considered as a single point. Direct measurement of traffic matrix requires placement of flow measurement at each ingress/egress points. This is impractical for large IP networks in terms of cost, time and effort [1]. As a result, several approaches have been implemented to estimate or model the traffic matrix from other available measurements. Link measurements provides traffic data for each link in a network. These measurements are easily collected from routers using SNMP (Simple network management protocol). In [2], authors proposed the method of network tomography, where used of traffic matrix is estimated from link measurements and routing information. The link and traffic matrix relationship has greater number of unknown than the number of equations. Thus, it cannot be solved for a unique solution. Approaches like Bayesian and Expectation Maximization model estimates the TM from statistical features of the traffic data [3][4]. These approaches take into account nature of traffic with changes in time. Spatial estimation of traffic matrix ignores the temporal features and works for a single time instance of TM matrix. Spatial model like Gravity model, Discrete choice model, independent connections etc. model have been implemented to produce TM. Different traffic model provides useful estimations based on the implementation. But these models are not actual representations and so they have inaccuracy. Several approaches were introduced where the output from these model are improved for higher accuracy. Traffic matrix estimation by using a combination of network tomography and spatial models was proposed in [5]. Estimation method in [6] implemented neural networks to improve TM based on expectation maximization model. Applying optimization methods to estimate and improve traffic matrix accuracy was proven successful in various studies [7, 8, 9]. In this paper we explore the use of GA as an optimization tool. Genetic Algorithm is a soft optimization technique which uses guided random techniques to search for an optimized solution. GA is known to work well for noisy environments and large parameter problems. For traffic matrix estimation using neural networks GA was applied to optimize the weights [10]. GA also found its implementation in Distributed Denial of Service attack. Parameters of traffic matrix were

*Department of Computer Science and Engineering, National Institute of Technology Silchar, India (anishkumarsaha@gmail.com)

optimized using GA to detect the attacks [11]. We proposed the combination of Gravity model and GA optimization to estimate and improve the accuracy of traffic matrix.

2. Related Works. Traffic Engineering (TE) deals with improvement of network performance by analysing traffic data and patterns. Traffic prediction is used in TE to control time varying traffic. Traffic variation are categorised into short-term and long-term variation. The long term variation helps to define the daily behaviour of traffic, while short term variation are handled to avoid congestion. Prediction is used to manage the traffic for optimal link utilization[12]. Dynamic behaviour of routing depends on proper prediction of traffic. In [13], the authors proposed optimization of router deployment depending on traffic flow to improve network efficiency. In their method, traffic flows are directed to a service node for making egress based optimization routing. The network paths are planned to facilitate an effective TE for improvement of quality of network traffic. In [14], a sequence-to-sequence model is proposed to improve the challenges caused by growth of Internet traffic. The sequence-to-sequence model learns forwarding path based on traffic data. In [15], authors investigated various TE techniques to improve different aspects of a network. They proposed Centralized Optimal Traffic Engineering system, Suboptimal Solution, and Distributed & Greedy solution to maximize data delivery. The performance of the network is measured in aggregated network throughput, average end to end delay and flow fairness. In [16], a heuristic approach is proposed to optimize routing over multiple TMs. The routes obtained through this methods are loop free and optimized bandwidth of links.

GA due to it's flexibility is implemented for different problems in networking. In [17], authors give weightage on quality of service for mixed traffic environment. They proposed a scheduling algorithm using GA for optimal resource allocation in a network. The quality of service is taken as fitness function in their proposed GA. The traffic is possible to be classified into different categories as shown in [18]. They used wavelet kernel function with GA for classification. This type of classification helps in intruder detection for unwanted traffics of various applications. There are other example of GA based traffic classifiers like, distance-based, K-Nearest Neighbors, and neural networks. Data points are separated using Mutual information, Dunn, and SD based biased measurement. It helps in Peer to Peer and non Peer to Peer traffic classification [19]. Classification of packets require a robust scheme that provide scalability, reliability and quality services. Feature selection, a classification process, selects relevant features during prediction. Both GA based classification and feature selection are used to classify packets as shown in [20]. An incomplete collection of traffic data degrade the integrity, information and quality. Fuzzy C-Means is an algorithm used to tackle clustering problem of incomplete traffic data. The missing data is filled up with the process called imputation, where estimated data replace the missed data. Fuzzy C-means with GA is making a good hybrid model for traffic data estimation[21].

Effective TE applications requires accurate estimation of Traffic matrix. A model named, network tomography equation establishes a relationship between link measurements and traffic matrix. This relationship is an under determined system. Due to this ill posed problem, numerous works focus on combining network tomography method with other mathematical models. The Gravity model construct TMs by assuming Origin-Destination flows proportional to the incoming and outgoing traffic of nodes. The generalised gravity further improves this process by classifying the traffic flows. The tomogravity model combines gravity and tomography model to increase accuracy of TM. Advanced tomogravity method introduces a relativity factor to further improve the estimation as in [22]. The co-variance method [23] uses co-variance matrix of link count sample to make up for insufficient information. TMs are estimated from the link count covariance matrix. This method provides a light weight estimation consistent with actual link measurements. The Generalized Autoregressive Conditional Heteroscedasticity model [24] deals with the ill posed nature of TMs. It provides a flexible approach to capture self similarity in traffic behaviour. Comparison of different TM estimation method [25] shows that tomogravity and entropy maximization performs better than linear programming approach. Adding extra constraints in entropy maximization further increase the performance. In [26], authors present the use of Simple Network Management Protocol for complete traffic collection with known estimation techniques to improve accuracy. The adaptive information gain maximization also focuses on traffic collection for improvement in accuracy. The most informative flow from traffic is determined for estimation of TMs. This approach increases the accuracy with a small increase in measurement resources [27]. For a large network, estimating TM takes high computation time for which division of network is one of the solutions. In [25], authors proposed divide and conquer method for estimating TM for large size network. The large network is divided into smaller

Table 3.1: An Instance of Traffic Matrix for duration of 5 min. on 01/03/2004

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}
S_1	10000	195828	615502	125898	154887	183703	136904	306701	169523	280277	145619	1178115
S_2	166931	30997781	6106169	1938388	1474276	10256961	1816513	6836966	4801483	533208	3359056	19405592
S_3	1422635	4775747	10779162	5151845	3686376	9884916	2451656	10415963	5286877	681353	1685721	3992645
S_4	86552	878056	14444321	1410746	3153286	2702903	1480837	8015838	2146278	5394549	3978549	4593323
S_5	89641	1108792	6334313	1384882	2102362	3428964	2673813	36921723	2724474	1105266	839099	3189994
S_6	1787597	3470525	45766716	8016824	12440169	8568396	3709065	9151891	15231037	1299895	5209570	15426785
S_7	157860	1666336	10114602	2022864	2053539	3006659	1508587	3252594	4815904	458907	916602	4518164
S_8	127373	7272971	33646381	3389768	3386575	16020094	5088844	4263424	22936657	866828	9569795	20897395
S_9	1461615	15332940	20127858	6129395	8995420	31247043	9287731	26633460	51298517	3596757	8225459	41947778
S_{10}	10000	390452	1892275	3902592	538756	1448104	786402	750079	829298	5351103	1236618	745390
S_{11}	41632	5955580	8442220	1627941	4238538	2884194	847818	6662390	9317015	1783127	389384	3944017
S_{12}	4207163	47226648	24952949	13523783	5789742	23543180	12241025	34378360	50123027	742716	11160076	70370056

parts then combined them for the full estimation of TM. This approach deals with smaller routing table and allows parallel processing which improves computational time. Artificial Neural Network approach is possible to improve estimation of TM. In [28], authors proposed Recurrent Neural Network approach for estimation of TM. The Recurrent Neural Network takes real world data for training of the model to extract spatio-temporal features of TM. Convolutional neural network is another finding for spatial relationship between link loads and Origin-Destination flows[29]. These artificial neural networks approach are good in finding hidden features of TM, although it requires large training data. TM estimation often includes aggregation of large traffic flows which is a difficult task at real time. In [30], authors proposed distributed Map Reduce approach for aggregation of large traffic flows. The Map Reduce approach uses topology information to identify origin-destination links, for easier aggregation of traffic flows. The input for the Map reduce is collected using a big data streaming module and the result shows near real time estimation of TM.

3. Proposed model for GA based Traffic Matrix Estimation.

3.1. Basics of Traffic Matrix Model. We have used a well known network for TM estimation named Abilene network collected by Y.Zhang, which contains data set of actual traffic matrix of the Abilene network for 6 months [31]. Each instance of TM collects 5 minutes of traffic between all source and destination. The routing matrix and topology are included in the data set. It consist of 12 nodes and 54 links. There are 32 uni-directional links between nodes and each nodes has an inbound and outbound link, therefore a total of 54 links. The TM of an instance X_i is a 2D matrix of 12×12 in size, an example is shown in Table 3.1. Each value x_{ij} in the matrix denotes the traffic flow from source S_i to destination D_j. With elapsed time t, the traffic matrix extend to a complete 3D traffic matrix as $X_1, X_2, X_3, ..., X_i$.

For determining TM X, two inputs are required namely, routing matrix A and link loads Y respectively. The relationship among them is as follows,

$$AX = Y \tag{3.1}$$

For this, all links load are measured for a network, from which different models are applied to have more precise TM values. In Abilene network, 54 numbers of link loads are available as seen 1D vector. Link loads are direct count of packets for a link. Abilene network has 144 (12×12) numbers of all possible combination of {Source, Destination} for all 12 nodes and 54 numbers of various links. Thus making the routing matrix A of $\{144 \times 54\}$ in dimension size. The entries of A are as follows,

$$A_{ij} = \begin{cases} 1, & \text{if link } i \text{ is utilized in } \{\text{Source-Destination}\}\ j \text{ path} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Eq. 3.1 is a linear system where the number of unknowns are larger than the number of equations. There are multiple solutions for the value of X. Proper use of mathematical models such like, Gravity model, provides a close estimation, yet further improvement is possible in the solution.

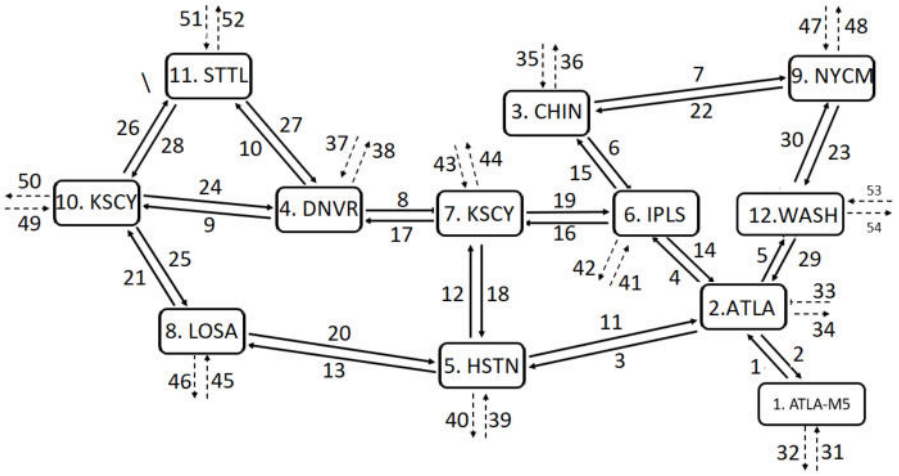


Fig. 3.1: Abilene network topology with various link traffic direction

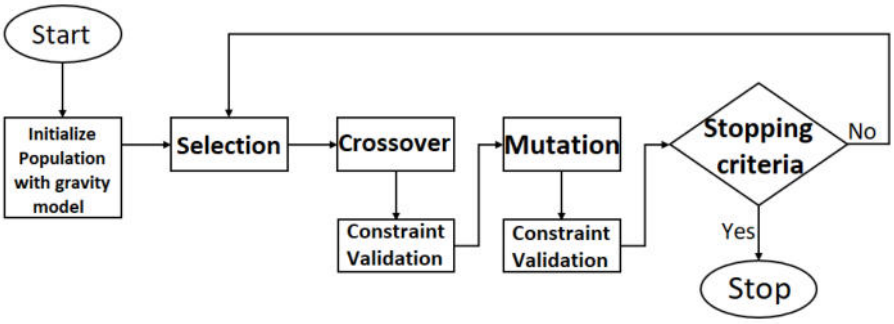


Fig. 3.2: Flowchart for proposed GA

3.2. Basics of Gravity Model. The Gravity model as the name suggest its an adaptation of Newton's Gravity model. This model is applied in different transportation problems, like road traffic, goods etc. In internet TM also it is applied to estimate traffic matrix by taking the ingress and egress flow of a node [32]. Each entry of the TM, X_{ij} represent the traffic from node i to node j. It is considered proportional to all traffic flowing out from i, T_i^{out} and all traffic flowing in to j, T_j^{in}. The total traffic $T^{total} = \sum_{k=1}^{n} T_k^{in} = \sum_{k=1}^{n} T_k^{out}$, is used for normalization. The expression for the TM entry is as follows:

$$X_{ij} = \frac{T_i^{out} \times T_j^{in}}{T^{total}} \tag{3.3}$$

where n is number of nodes in the network and the output is a traffic matrix of X$\{n \times n\}$ dimension, where $1 \leq i \leq n$ and $1 \leq j \leq n$.

3.3. Proposed TM model. The Concept of GA, was introduced as an optimization technique that models after the evolution of biological being in the natural world. GA is a heuristic random search that finds sub-optimal solution. The first step of GA is initialization, where a set of random feasible solution for the optimization is generated. GA basic operators, like crossover and mutation are applied to form new population. The best solutions are selected from the population. The operations are repeated until the result converged towards a sub-optimal solution [33]. The gravity model solution is possible to further optimize using GA to

closer match the link measurements. The optimization with the help of Eq. 3.1 is defined as follows:

$$\begin{array}{lll} \text{Min.} & \|AX - Y\| + w\,\|X - X_{GM}\| & \\ \text{Subject to} & \sum_{j=1}^{n} x_{ij} = I_i & \\ & \sum_{i=1}^{n} x_{ij} = J_j & \\ & x_{ij} \geq 0 & i = 1...n \;\&\; j = 1...n. \end{array} \tag{3.4}$$

where, I_i and J_j are the total incoming and outgoing traffic for node i and node j respectively. The first part of the objective function gives the distance of calculated link load from the link measurement, while the second part calculates the TM solution distance from the Gravity model. The w is a weight applied to the distance with gravity model in order to adjust the significance of Gravity model in the optimization. First and second constraint maintains the conservation of traffic, i.e. total traffic is constant. Third constraint makes sure that the traffic between any two nodes is always positive. The details of the proposed GA operators, shown in Fig. 3.2, for optimization of Eq. 3.4 are discussed below.

3.3.1. Initialization. We need a set of random solution known as initial population. For making initial population, a first individual is generated from Gravity model. Then after the remaining population are generated by making random changes to the first individual while maintaining all constraints. For this we introduce a matrix Δ of dimension $\{n \times n\}$ which is defined as:

$$\Delta = \begin{bmatrix} [B] & -[B] \\ -[B] & [B] \end{bmatrix}. \tag{3.5}$$

B is a matrix of size $(\frac{n}{2} \times \frac{n}{2})$, where each element b_{ij} takes a random value from the set {-1, 1}. Therefore Δ contains random elements of 1 or -1, and due to its arrangement given in (1), the sum of each row and column is equal to 0. A random multiplier, r is generated to calculate random solution. If X_{GM} represents the TM obtained by gravity model, then the rest of individual can be calculated as,

$$X = X_{GM} + r\Delta \tag{3.6}$$

Eq. 3.6 is possible to evaluate with multiple repetition for different values of r for generating multiple individuals. Since the sum of rows and columns of Δ is 0, the new individuals satisfy the first and second constraint.

3.3.2. Selection. Individuals for crossover are selected using the selection operator. For this purpose a roulette wheel is implemented. Roulette wheel as the name suggest takes inspiration from the game. The individuals are placed on the wheel as a pie chart structure. The fitter individuals occupy larger space on the wheel. A fixed pointer is placed on the wheel. When the wheel is spun the individual that the pointer lands on is selected. The roulette wheel provides a random selection while giving preferences to the better solution.

3.3.3. Crossover. The crossover takes two individuals and exchange information between them to form new individuals or offspring. Two individuals P and Q are selected as parents where P is the fitter parent and the generated offspring are C and D. A random gene location with index i and j is selected to apply crossover as below,

$$\begin{array}{l} c_{ij} = p_{ij} + \beta \times (p_{ij} - q_{ij}) \\ d_{ij} = q_{ij} + \beta \times (p_{ij} - q_{ij}) \end{array} \tag{3.7}$$

where β is a random number in the range of $[0, b]$. The remaining gene locations in C and D are directly copied from P and Q respectively. This crossover is known as the heuristic crossover [34]. Other crossovers like, average, arithmetic, blend etc., are also applicable in this process. The heuristic crossover is suitable here for finding a new solution closer to the fitter parent and the range of values for child genes are adjustable with b. The same is shown in Fig. 3.3.

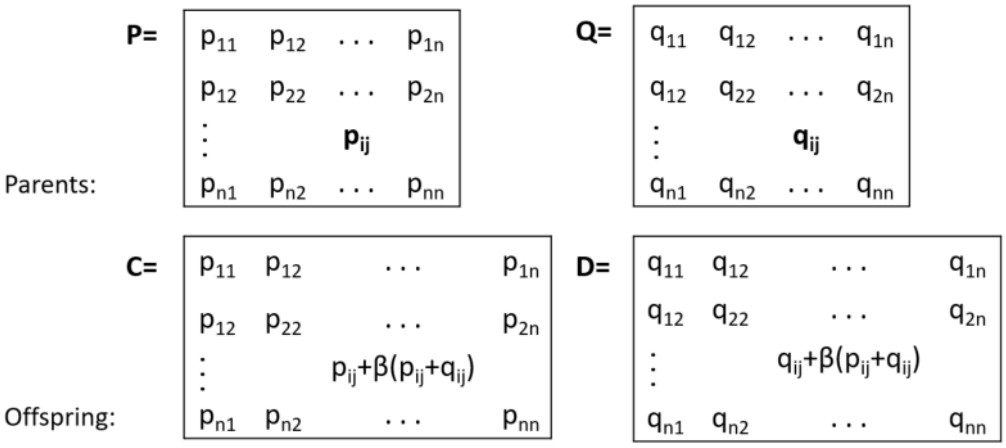


Fig. 3.3: Crossover operation

3.3.4. Mutation. The Mutation makes random change in gene values. This process is generally used to prevent the solutions converging to a local optimum point or premature convergence. For an individual P mutation is performed to produce one offspring C as follows,

$$c_{ij} = \alpha \times p_{ij} \tag{3.8}$$

where α is a random number in the range of $[0, a]$. The gene location with index i and j are randomly selected. The rest of the gene remain unchanged. Here, the rate of mutation is kept to 0.2.

3.3.5. Constraint Validation. The crossover and mutation when applied to a single gene value leads to violation in first and second constraints. These violation of constraints are handled using direct method as explained in [35]. The offspring of crossover and mutation are processed for constraint validation. If x'_{ij} and x_{ij} denotes the old and updated values for offspring X, the change in gene value is calculated as $\delta = x_{ij} - x'_{ij}$. Changes are made in X as follows:

$$x_{kl} = \begin{cases} x_{kl}, & \text{if } i = k \text{ and } j = l \\ x_{kl} + \frac{\delta}{n+1}, & \text{if either } i = k \text{ or } j = l \\ x_{kl} + \frac{\delta}{(n+1)^2}, & \text{otherwise} \end{cases} \tag{3.9}$$

where $k = 1 \ldots n$ and $l = 1...n$. This maintains the traffic conservation of first two constraints. Constraint 3 is handled by assigning a penalty to the fitness function when constraint is violated.

4. Results and Analysis. The results obtained from Gravity model and the proposed GA for the same traffic are compared. The traffic is collected from Abilene network available in [31]. The error is calculated using Root Mean Square Error(RMSE). The RMSE is calculated as follows,

$$RMSE = \sqrt{\frac{1}{n}\left(X_{estimated} - X_{raw}\right)^2} \tag{4.1}$$

Population size plays an important role in the proposed model. The algorithm is executed for different population size for the same time instance $t = 1$ as shown in Figure 4.1. It is observed that above 300, almost the RMSE is reaching to lowest value. A population size of 300 is taken as the optimal size. Figure 4.2 shows the comparison of TM values for the proposed GA and Tomogravity proposed in [5]. We observe that the GA estimation are closer to the real value, which is represented as a diagonal line. The RMSE and average error comparison is shown in Table 4.2. It is observed that the RMSE value shows good improvement while

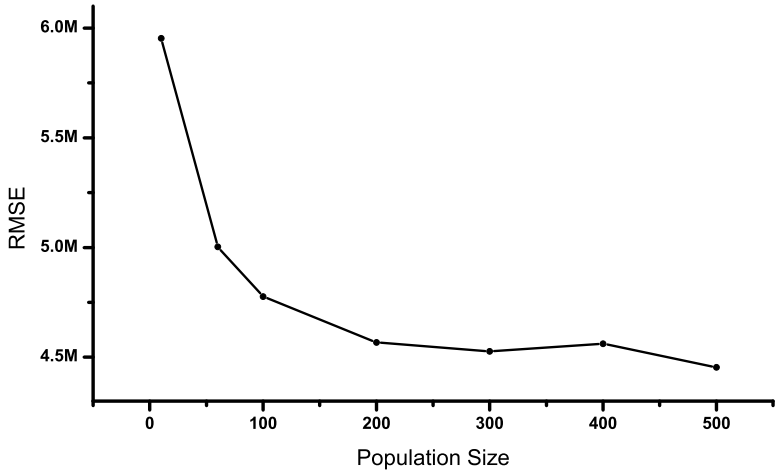


Fig. 4.1: Output of proposed GA with Population size for time instance, t=1.

Table 4.1: Effects of parameters in GA operators.

(a) Effect of α range in mutation.

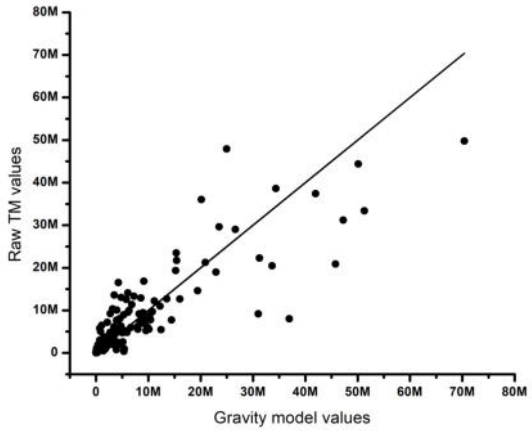
Value of a	**Generation number**
0.25	417
0.5	344
1	247
2	253
4	280
8	329
16	365

(b) Effect of β range in crossover.

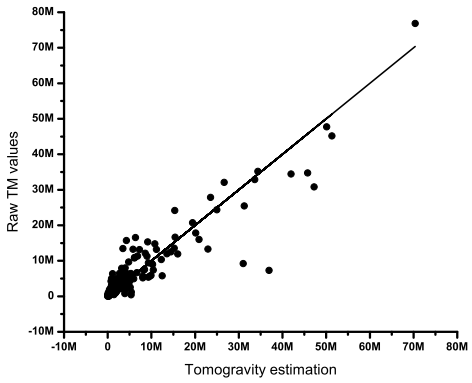
Value of b	**Generation number**
1	678
2	337
4	291
8	240
16	237
32	251
64	287

the average error improvement is lesser. This shows that the proposed GA estimation give higher accuracy for large traffic flows which contribute more to the entire traffic. The TM estimation for time instances $t = 1$ to 50 shows that the proposed GA provides improved results over the initial Gravity model and the Tomogravity results as seen in Figure 4.3.

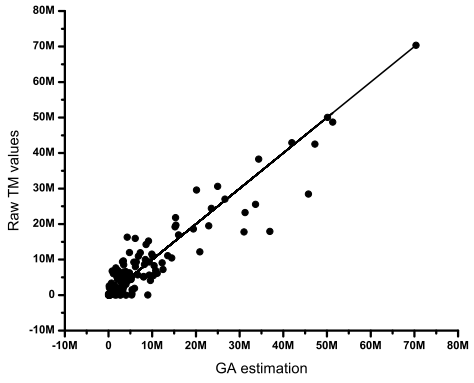
The effectiveness of mutation and crossover decides the performance of GA. Varying the range of the random multiplier α and β from the range $[0, a]$ and $[0, b]$ affects the performance of the proposed GA in mutation and crossover respectively. The optimal value of the proposed GA for different values of a and b are shown in Table 4.1. The proposed algorithm is executed with different values of a and b, and the outcomes are measured in the number of generation to converge to an optimized solution. For mutation Table 4.1a, $a = 1$ shows the best result, obtaining the solution at 247 generations. For crossover Table 4.1b, $b = 16$ shows the best result, obtaining the solution at 237 generations.



(a) Gravity



(b) Tomogravity



(c) GA estimation

Fig. 4.2: Comparison of various TM to raw value for instance t=1.

Table 4.2: Comparison of RMSE and Averrage error TM instance t=1.

	Gravity Model	**Tomogravity**	**GA estimation**
RMSE	6.1510×10^6	4.7038×10^6	4.2787×10^6
Average error	3.4572×10^6	2.6660×10^6	2.7638×10^6

5. Conclusion. Traffic matrix calculation for large IP network is a difficult task due to insufficient information. In this paper we have proposed a way to improve estimation for traffic matrix. The gravity model provides a reasonable estimation but can be improved with GA optimization. Higher population size provides a more diverse population thus allowing for better exploration of the search space. As the TM consist of 144 elements, such a high dimensional search space requires a larger population size. The GA estimation gives reasonable result at population size beyond 100, and as it further increases beyond 300 no significant improvement

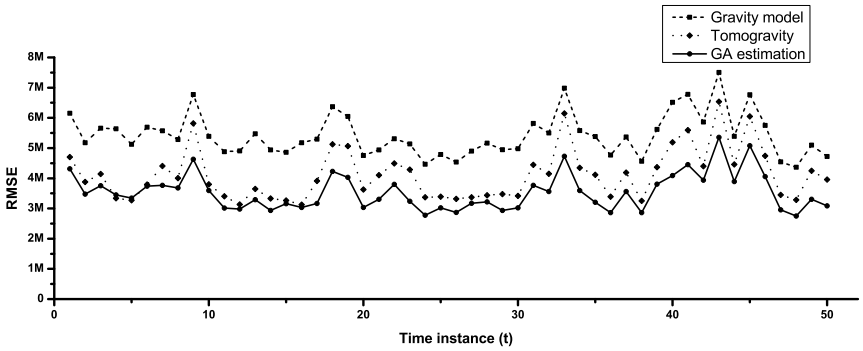


Fig. 4.3: RMSE of Gravity model and proposed GA for different time instances t= 1 to 50 & fixed population size= 300.

is observed. Our proposed model shows improvement over the gravity model in both RMSE and average error, while comparison with tomogravity results shows improvement in RMSE value shows better improvement than the average error. This shows that the propose GA estimation is more sensitive to the larger traffic values which are more significant to the overall matrix. Overall the proposed model provides improvement for traffic matrix model and theoretically it can be used with any TM model by replacing the Gravity model for an improved result.

REFERENCES

[1] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[2] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American statistical association*, vol. 91, no. 433, pp. 365–377, 1996.

[3] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 557–573, 1998.

[4] J. Cao, D. Davis, S. Vander Wiel, and B. Yu, "Time-varying network tomography: router link data," *Journal of the American statistical association*, vol. 95, no. 452, pp. 1063–1075, 2000.

[5] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 206–217, 2003.

[6] H. Zhou, L. Tan, Q. Zeng, and C. Wu, "Traffic matrix estimation: A neural network approach with extended input and expectation maximization iteration," *Journal of Network and Computer Applications*, vol. 60, pp. 220–232, 2016.

[7] S. Eum, J. Murphy, and R. Harris, "A fast accurate lp approach for traffic matrix estimation," in *International Teletraffic Congress ITC19*, 2005.

[8] O. Goldschmidt, "Isp backbone traffic inference methods to support traffic engineering," in *Internet Statistics and Metrics Analysis (ISMA) Workshop*, pp. 1063–1075, 2000.

[9] D. Jiang, X. Wang, and L. Guo, "An optimization method of large-scale ip traffic matrix estimation," *AEU-International Journal of Electronics and Communications*, vol. 64, no. 7, pp. 685–689, 2010.

[10] A. Omidvar and H. Shahhoseini, "Intelligent ip traffic matrix estimation by neural network and genetic algorithm," in *2011 IEEE 7th International Symposium on Intelligent Signal Processing*, pp. 1–6, IEEE, 2011.

[11] S. M. Lee, D. S. Kim, J. H. Lee, and J. S. Park, "Detection of ddos attacks using optimized traffic matrix," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 501–510, 2012.

[12] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, and K. Shiomoto, "Traffic prediction for dynamic traffic engineering," *Computer Networks*, vol. 85, pp. 36–50, 2015.

[13] J. Sun, S. Sun, K. Li, D. Liao, A. K. Sangaiah, and V. Chang, "Efficient algorithm for traffic engineering in cloud-of-things and edge computing," *Computers & Electrical Engineering*, vol. 69, pp. 610–627, 2018.

[14] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Generation*

Computer Systems, vol. 92, pp. 59–67, 2019.
[15] M. Islam, M. A. Razzaque, M. Mamun-Or-Rashid, M. M. Hassan, A. Alelaiwi, and A. Alamri, "Traffic engineering in cognitive mesh networks: Joint link-channel selection and power allocation," *Computer Communications*, vol. 116, pp. 212–224, 2018.
[16] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in hybrid sdn networks with multiple traffic matrices," *Computer Networks*, vol. 126, pp. 187–199, 2017.
[17] S. H. da Mata and P. R. Guardieiro, "Resource allocation for the lte uplink based on genetic algorithms in mixed traffic environments," *Computer Communications*, vol. 107, pp. 125–137, 2017.
[18] F. Ertam and E. Avci, "A new approach for internet traffic classification: Ga-wk-elm," *Measurement*, vol. 95, pp. 135–142, 2017.
[19] M. Mohammadi, B. Raahemi, A. Akbari, H. Moeinzadeh, and B. Nasersharif, "Genetic-based minimum classification error mapping for accurate identifying peer-to-peer applications in the internet traffic," *Expert Systems with applications*, vol. 38, no. 6, pp. 6417–6423, 2011.
[20] J. Park, H.-R. Tyan, and C.-C. J. Kuo, "Ga-based internet traffic classification technique for qos provisioning," in *2006 International Conference on Intelligent Information Hiding and Multimedia*, pp. 251–254, IEEE, 2006.
[21] J. Tang, G. Zhang, Y. Wang, H. Wang, and F. Liu, "A hybrid approach to integrate fuzzy c-means based imputation method with genetic algorithm for missing traffic volume data estimation," *Transportation Research Part C: Emerging Technologies*, vol. 51, pp. 29–40, 2015.
[22] H. Zhou, L. Tan, F. Ge, and S. Chan, "Traffic matrix estimation: Advanced-tomogravity method based on a precise gravity model," *International Journal of Communication Systems*, vol. 28, no. 10, pp. 1709–1728, 2015.
[23] I. Juva, S. Vaton, and J. Virtamo, "Quick traffic matrix estimation based on link count covariances," in *2006 IEEE International Conference on Communications*, vol. 2, pp. 603–608, IEEE, 2006.
[24] D. Jiang and G. Hu, "Garch model-based large-scale ip traffic matrix estimation," *IEEE Communications Letters*, vol. 13, no. 1, pp. 52–54, 2009.
[25] M. M. Rahman, S. Saha, U. Chengan, and A. S. Alfa, "Ip traffic matrix estimation methods: Comparisons and improvements," in *2006 IEEE International Conference on Communications*, vol. 1, pp. 90–96, IEEE, 2006.
[26] A. Gunnar, M. Johansson, and T. Telkamp, "Traffic matrix estimation on a large ip backbone: a comparison on real data," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 149–160, 2004.
[27] D. Li, C. Xing, N. Dai, F. Dai, and G. Zhang, "Estimating sdn traffic matrix based on online adaptive information gain maximization method," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 465–480, 2019.
[28] J. Zhao, H. Qu, J. Zhao, and D. Jiang, "Towards traffic matrix prediction with lstm recurrent neural networks," *Electronics Letters*, vol. 54, no. 9, pp. 566–568, 2018.
[29] M. Emami, R. Akbari, R. Javidan, and A. Zamani, "A new approach for traffic matrix estimation in high load computer networks based on graph embedding and convolutional neural network," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, p. e3604, 2019.
[30] W. J. Queiroz, M. A. Capretz, and M. A. Dantas, "A mapreduce approach for traffic matrix estimation in sdn," *IEEE Access*, vol. 8, pp. 149065–149076, 2020.
[31] Y. Zhang, *Abilene Dataset, https://www.cs.utexas.edu/ yzhang/research/AbileneTM/*, 2004.
[32] M. Roughan, "Simplifying the synthesis of internet traffic matrices," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 93–96, 2005.
[33] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.
[34] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of genetic algorithms*, vol. 1, pp. 205–218, Elsevier, 1991.
[35] B. Craenen, A. Eiben, and E. Marchiori, "How to handle constraints with evolutionary algorithms," *Practical Handbook Of Genetic Algorithms: Applications*, pp. 341–361, 2001.

Edited by: Dana Petcu
Received: Nov 13, 2020
Accepted: Jan 21, 2021


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 39–52, DOI 10.12694:/scpe.v22i1.1836

A MICROSERVICE DECOMPOSITION METHOD THROUGH USING DISTRIBUTED REPRESENTATION OF SOURCE CODE

OMAR AL-DEBAGY* AND PÉTER MARTINEK

Abstract. This research proposed a novel decomposition method for refactoring monolithic applications into microservices applications using a neural network model (code2vec) for creating code embeddings from the monolithic application source code. As a Result, semantically similar code embeddings are clustered through a hierarchical clustering algorithm to produce microservices candidates to resemble the domain model more efficiently. The quality characteristics of the results were measured using two metrics for measuring cohesion. These metrics were Cohesion at Message Level (CHM) and Cohesion at Domain Level (CHD). Also, four applications were used as test cases with different sizes ranging from small to big applications. The proposed method showed promising results in terms of cohesion when compared to other decomposition methods. The proposed method scored better scores in 5 out of 8 tests compared to other methods. Also, averaged CHD and CHM results were 0.52 and 0.76, respectively, for the proposed method, better results when compared to the other methods.

Key words: microservices decomposition, microservices, refactoring

AMS subject classifications. 68M14

1. Introduction. Nowadays, the internet requires a more flexible, scalable, and understandable software architecture. Therefore, many companies and organizations started the process of migration from the monolithic architecture toward a more suitable architecture that meets the demands of the current market [1]. The current market requires an architecture flexible enough to face the frequent changes in user demands, and easily scalable architecture to face the massive number of users [2]. These reasons led many companies and organizations to adopt the microservices architecture. They chose microservices architecture to have a more scalable, easier to maintain, and easier to manage applications [3]. Microservices is an architecture of fine-grained services that are working independently from each other and communicating with each other through lightweight mechanisms to do the tasks of an application suite. Although microservices provide different advantages to the organization, but the process of migration introduced multiple issues. One of these issues is how to decompose or refactor an existing monolithic application into microservices, which are loosely coupled and highly cohesive at the same time, according to Newman [4]. Multiple researchers have introduced several methods to decompose the monolithic application [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. These methods include many different approaches such as static analysis of the application's code [8, 9, 10, 14], dynamic analysis of the performance of the applications [11, 12, 15, 16], and analysis of the applications interfaces [7, 17, 13]. These methods provide assistance for developers to help them in the process of migrating from monolithic applications to the microservices application. Still, the result of these methods cannot be considered as absolute results. This research aims to tackle the issue of microservices identification using vector representation of software's source code. Hence, this paper proposed a novel approach to decompose monolith application into a microservices application by using a neural model [22] to represent snippets of code as continuous distributed vectors. The code of the monolithic application would be converted into vector representation using the provided model, and then certain classes would be grouped to provide microservices candidates.

Four monolithic applications with different sizes were decomposed using the provided methodology in order to verify the effectiveness of this approach. These applications were tested in other research papers before, so they are considered benchmark applications for monolithic applications' decomposition process. We used two different metrics to compare the proposed method's performance with other methods from the literature. Also,

*Department of Electronics Technology, Budapest University of Technology and Economics, Budapest, Hungary (omeraldebagy@gmail.com)

we utilized other metrics to compare the sizes of the mentioned applications in the test, such as the number of classes, number of methods, lines of code (LoC), and the number of microservices. The proposed method is a useful aiding tool for developers in the process of migration from monolithic to a microservices architecture, which suggests a specific direction for the decomposition process.

The goal of this paper is to investigate the effectiveness of using code embeddings in the process of decomposing monolithic applications into microservices ones. Also, provide a new technique for extracting microservices from monolithic applications.

The rest of this research is organized as follows: the literature review provides different approaches for handling microservices decomposition. Then, the methodology section presents the details of the proposed methodology and how the decomposition method was constructed. After the methodology, there is a section with the results and the discussion. Finally, a conclusion section highlights future potentials for the method and other possible implementations.

2. Literature Review. There are several research papers related to microservices identification or decomposition. These researches provide different types of methods that can be divided into three groups. The first group is based on the static analysis of the source code of the monolithic application. The second group is based on the dynamic analysis of the monolithic application. Finally, the third group is using the analysis of the application program interfaces or application interfaces.

Abdullah et al. [15] created a decomposition method that considers the scalability and performance of the application and improve its performance after decomposition. Their method used an unsupervised machine learning approach analysing access logs of monolithic applications. Then, they proposed a method to automatically assign the type of virtual machines and their resources to the microservices instances on a cloud architecture. Their method of decomposing a monolithic application based on the application's performance can be misleading because it depends on how the application can be used or how the users are using it. Therefore, the methods based on performance analysis need to have very detailed testing scenarios to work efficiently.

Mazlami et al. [9] proposed a decomposition method for monolithic applications by analysing the version control repository of the application and converting it into graphs for detecting microservices candidates using a graph clustering algorithm. Their method consisted of three different extraction strategies, which are Logical Coupling Strategy, Semantic Coupling Strategy, and Contributor Coupling Strategy. One limitation of the method is the use of classes without considering methods and their input and output parameters.

Kamimura et al. [8] created a method for extracting microservices candidates from source code using a clustering algorithm. They tested their method on two different applications, and two developers reviewed their results. Also, they visualized the provided microservices for the ease of understanding with the Software Architecture Finder (SArF) map for visualization.

Li et al. [6] proposed a data-flow driven approach for decomposing monolith applications into microservices candidates. They highlighted how the decomposition process is different between service-oriented architecture (SOA) and Microservices. First, services in SOA are coarse-grained while in microservices are fine-grained. Second, the process is bottom-up in SOA and top-down first then bottom-up in microservices. Their method consists of 4 steps, first analysing requirements of the monolithic application. Second, constructing data flow diagrams (DFD). Third, compress DFDs into decomposable DFDs. Fourth, propose microservice candidates through decomposable DFDs.

Furthermore, they used cohesion and coupling metrics to evaluate their results compared to Service Cutter and API analysis. The issue with this approach is the need for attention to details in order to create a detailed DFD to make the process of identifying appropriate microservices. Furthermore, they used a relatively small application for the evaluation.

Taibi and Systa [11] proposed a decomposition method using a data-driven approach based on process mining by utilizing log files as a data source. Their decomposition method consisted of 6 steps. The first step is the execution analysis path; the second step is the frequency analysis of the execution path. Removing circular dependencies is the third step. The fourth step is identifying decomposition options. The fifth step is ranking the decomposition options based on metrics. Finally, the sixth step is selecting the decomposition option. They used coupling and the number of classes as metrics for step five. Their evaluation method of depending on the coupling metric is lacking because their method needs other metrics such as cohesion.

Table 2.1: Literature Review Summary

Research	Input	Decomposition Method	Year	Tested Application
Abdullah et al. [15]	Log Files	Performance Based	2019	ACME Air
Mazlami et al. [9]	Commits	Version Control Analysis with Graphs	2017	Multiple
Kamimura et al. [8]	Source Code	SArF software clustering algorithm	2018	PetClinic
Li et al. [6]	Data Flow Diagrams	Analyzing DFD	2019	Cargo App
Taibi and Systa [11]	Log Files	Analysis of the execution paths	2019	Industrial App
Saidani et al. [14]	Source Code	nondominated sorting genetic algorithm	2019	JPetstore, Spring Blog
Jin et al. [12]	Log Files		2018	JPetstore, SpringBlog, JForum, Roller
Nunes et al. [10]	Source Code	Clustering call graphs	2019	LdoD, Blended Workflow
Service Cutter [17]	System Specification Artifacts	Clustering of graphs	2016	Cargo App
Al-Debagy and Martinek [7]	API	Clustering of Operations	2019	Cargo App, Kanban Board, Money App
Santos and Silva [5]	Method Calls	Clustering call graphs	2020	LdoD, Blended Workflow, FenixEdu

Saidani et al. [14] introduced a new decomposition method called MSExtractor. They used the source code of monolithic applications to extract classes and group classes to create microservices candidates. For the evaluation of their method, they used cohesion and coupling metrics. Furthermore, they are utilized a non-dominated sorting genetic algorithm to identify microservices from the source code. This research is compared to the proposed method of this research paper.

Jin et al. [12] proposed a functional oriented decomposition method for microservices applications that monitor the application dynamic behaviour and clusters execution logs or traces. They proposed some evaluation metrics for cohesion and coupling. The logs are generated using specific test cases, but sometimes these test cases cannot cover all the business functionalities, which may lead to ignoring some essential classes and functionalities. The metrics proposed in Jin et al. are used in this paper in order to compare the performance of the proposed method to other decomposition methods.

Nunes et al. [10] developed a method that converts the source code of a monolithic application into call graphs. After that, domain entities are identified, and a clustering algorithm will group these entities. This work has several limitations such as, it is developed for a specific web application framework, and the tool that creates call graphs does not work correctly with all Java versions.

Service Cutter [17] is a decomposition tool that uses domain models and use cases to extract coupling information. This coupling information was defined by the authors and was represented as a weighted graph using Epidemic Label Propagation clustering algorithms.

Al-Debagy and Martinek [7] introduced a decomposition method that relies on the monolithic application's API. They used API operation names to identify the microservices through grouping semantically similar operation names using a hierarchical clustering algorithm.

Santos and Silva [5] proposed a decomposition method that collects graph calls of the monolithic application and converts them into domain entities. Then a similarity function measures the similarity between two entities, and a clustering algorithm groups similar entities together to create microservices candidates. Also, they proposed a complexity metric to verify the validity of the suggested microservices candidates.

Table 2.1 summarizes the methods mentioned in the literature review section 2 and included the applied types of inputs and the type of decomposition they used.

3. Methodology. Machine learning for code refactoring was used on several other software architectures before [23, 24, 25]. However, it can be applied in a microservices' environment as well. This research proposes a new decomposition method for decomposing monolithic applications into microservices applications as follows. The approach uses a novel approach for microservice decomposition by using code representation to understand the similarity within the application classes and cluster semantically similar classes together to create microservices candidates. Clustering semantically similar classes together in order to resemble the domain model more efficiently.

The proposed machine learning-based method consists of these steps:

1. extracting the methods and its code from the monolithic application,
2. converting the code to code embeddings or vector representations,
3. aggregating the code embeddings of one class,
4. group together semantically similar classes to obtain microservices candidates.

3.1. Extracting Code Embeddings. Methods are extracted from classes and converted into code embeddings using the code2vec [22] model. Code embeddings are snippets of codes characterized as a vector-based representation for a machine-learning algorithm to understand these snippets of codes.

Embeddings are a mapping of an object represented as vectors. For example, word embeddings are representations of a word (or sequence of words) as vectors of real numbers [26]. Word embeddings make it possible for textual data to work with a mathematical model. Code embeddings have a similar benefit to word embeddings; these embeddings can capture the semantics of the source code. These code embeddings can be used for several tasks such as malware detection, author identification, and refactoring.

3.2. Code Embeddings Model. The proposed method uses the code2vec model created by Alon et al. [22] to obtain code embeddings or continuous distributed vectors of the extracted methods. Code embeddings give us the ability to find a similarity between the extracted classes.

Code2vec is a deep representation learning method, which was used for predicting method names. However, code2vec code embeddings can be used in other tasks as well. Code2vec converts the source code into a set of Abstract Syntax Tree (AST) paths and sums these paths using an attention mechanism. The attention technique works by giving more weight for the important AST paths that represent the source code. So, the vector representation of a function is an aggregation of weighted AST paths. The attention mechanism shows the important AST paths that need more focus than the other available paths.

AST is represented with branches and leaves similar to a tree. The functional structure of source code is represented by AST instead of a detailed description of source code. For example, Fig. 3.1 shows an AST representation of a factorial function. The utilization of AST improve the accuracy and training of a machine learning model [26].

The goal of code2vec is to generate code embeddings that keep the semantics of the source code. Code2vec represent the source code as a bag of AST paths; these paths are generated between the leaves of the AST tree. AST path is a path between two leaves in an AST tree. For example, the coloured paths in Fig. 3.1 are AST paths. Path-context is a set of three tokens, consisting of two tokens represent the two AST leaves and another token represent the path between these two leaves. For example, the red path in Fig. 3.1 can be represented as follows:

$$\{n, Times \downarrow MethodCall \downarrow Minus \downarrow, n\}$$

The sign $\downarrow$ represent the path goring toward the leaves while $\uparrow$ represent going toward the root of the AST tree. For more details and information check the original paper [22]. Fig. 3.2 shows the architecture of code2vec model with all the processes described earlier.

3.3. Aggregation Method. This step combines the code embeddings of the methods in order to reflect the representation of the class. Multiple aggregation functions were used, such as mean, sum, maximum, minimum, standard deviation, and variance. The mean function gave the best results regarding the accuracy of the clustering function in the next step. Fig. 3.3 shows the process of aggregating multiple code embeddings into one vector representation using the mean function. Table 3.1 lists all the aggregation methods that we tested to find the most applicable aggregation method for the proposed decomposition algorithm.

After this step, the aggregated code embeddings are sent to the next step, which is the clustering method, where it will generate the microservices candidates.

3.4. Clustering Method. Following the conversion of the source code into code embeddings based on code2vec model and aggregating code embeddings, a clustering method was applied. Related classes are clustered together using the clustering method in order to generate a suitable microservice candidate. The Affinity Propagation [19] algorithm was chosen for this process because it identifies the sum of clusters minus the necessity to indicate it in advance. Microservices candidates are identified using the previously mentioned methods combined with the clustering algorithm.

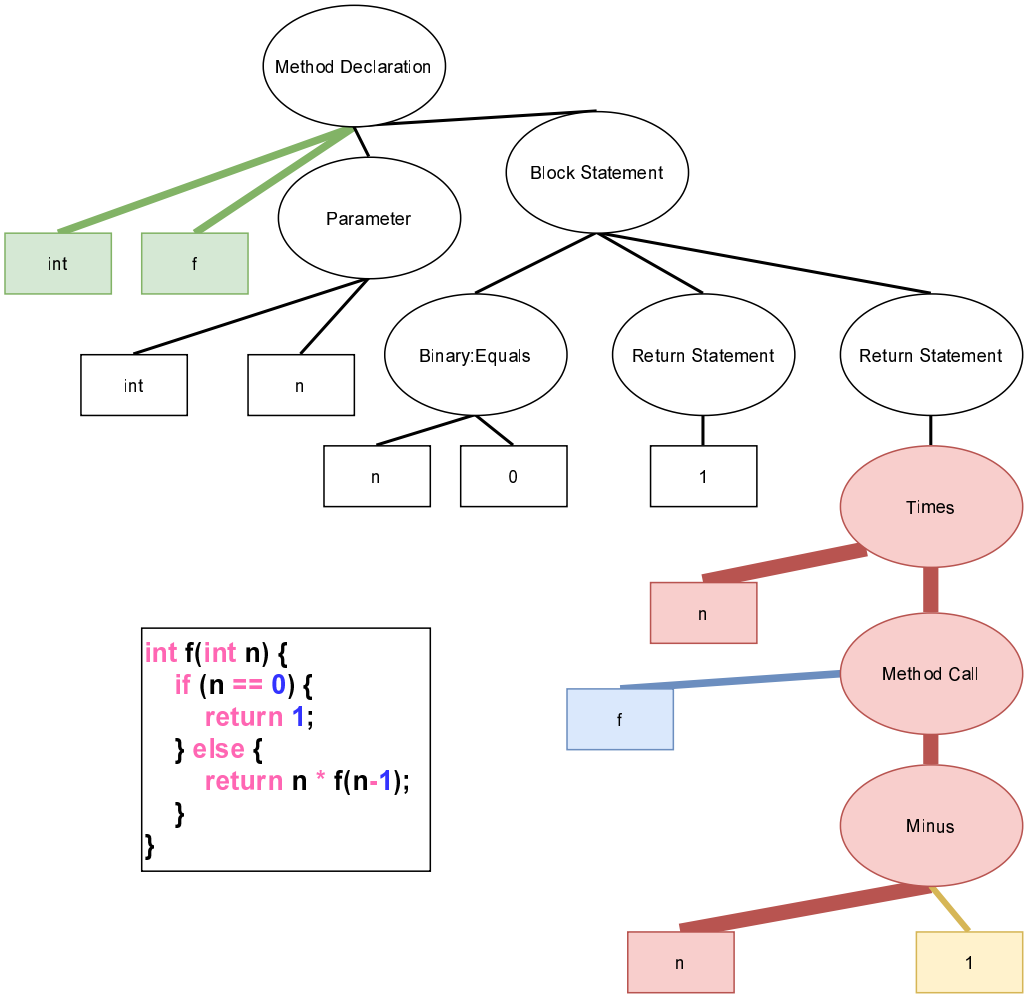


Fig. 3.1: AST representation of a factorial function

The Affinity Propagation algorithm is based on two concepts that are passing messages between data points and finding exemplars [19]. Exemplars are the centres of each cluster, which represent the cluster, and each cluster contains a single exemplar. Also, there are two types of these exchanged messages between the data points. The first type is exchanged between the data points and the candidate exemplars, and these types of messages are called (responsibility) messages. They are used to find the strength of the link between the data points and the exemplars.

The (responsibility) messages are represented by $r(i,k)$ in equation 3.1 implies if point k is fit to be an exemplar for point i. Responsibilities are exchanged from point i to exemplar to be k:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \text{ s.t. } k' \neq k} \{a(i,k') + s(i,k')\} \tag{3.1}$$

The second type checks the suitability of an exemplar in being an exemplar by sending messages from the exemplar candidates to other data points in the cluster. This type of messages referred to as (availability) messages. The (availability) represented by $a(i,k)$ in equation 3.2 shows if point i can select point k as an exemplar. Availabilities are exchanged between exemplar candidate k and data point i starting from k:

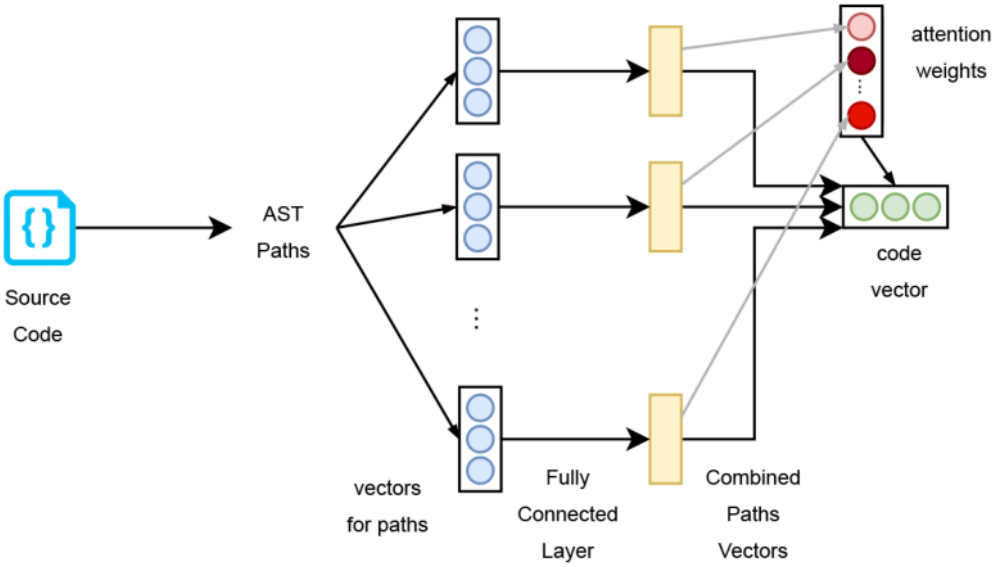


Fig. 3.2: code2vec Model [22]

Table 3.1: Aggregation Methods

Aggregation Method	Equation
Mean	$\frac{1}{n}\sum_{i=1}^{n} x_i$
Summation	$\sum_{i=1}^{n} x_i$
Maximum	$x_i : x_i \geq x_i, i \neq j \forall i, j \in n$
Minimum	$x_i : x_i \leq x_i, i \neq j \forall i, j \in n$
Standard Deviation	$\sqrt{\frac{\sum(x_i - \overline{x})^2}{(n-1)}}$
Variance	$\frac{\sum(x_i - \overline{x})^2}{(n-1)}$

$$a(i,k) \leftarrow \min\left\{0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\} \tag{3.2}$$

Equation 3.3 shows the method of updating self-availability for an exemplar:

$$a(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\} \tag{3.3}$$

Then pairwise similarities are used to identify the similarities between the data points. Also, clusters can be found by maximizing the total similarity between the exemplars and their data points.

Mezard [20] described the significance and effectiveness of message passing algorithms, even on complex problems. Thus, Affinity Propagation was used for our research paper for clustering related classes to generate microservices candidates.

Affinity Propagation algorithm includes three parameters which affect the performance of the clustering algorithm:

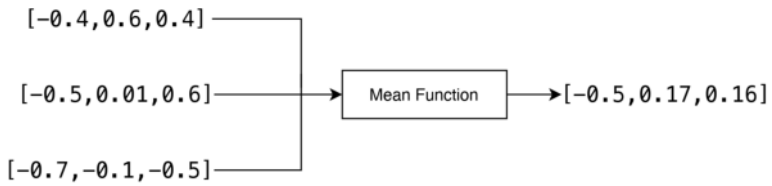


Fig. 3.3: Mean Aggregation Method

1. The first parameter is damping, which checks the interchange of messages between responsibility and availability to avoid numerical fluctuations while updating the values of responsibilities and availabilities [21].
2. The second parameter is the maximum number of iterations.
3. The third one is the number of iterations with no change in the number of estimated clusters that stop the convergence.

Algorithm 1 shows the steps of the Affinity Propagation algorithm.

Algorithm 1 Affinity Propagation algorithm

Input: $\{s(i,j)\}_{i,j\in\{1,\ldots,N\}}$ data similarities and preferences
Output: cluster assignments $\hat{c}$
Availability$\leftarrow$0
repeat a and r updates until convergence
  $r(i,k) \leftarrow s(i,k) - \max_{k' \text{s.t.} k'\neq k}\{a(i,k') + s(i,k')\}$
  $a(i,k) \leftarrow \min\left\{0, r(k,k) + \sum_{i' \text{ s.t. } i'\notin\{i,k\}} \max\{0, r(i',k)\}\right\}$
  if $k \neq i$ **then**
    $a(k,k) \leftarrow \sum_{i' \text{ s.t. } i'\neq k} \max\{0, r(i',k)\}$
  end if
until convergence
Return $\hat{c} = argmax_k\,[a(i,k) + r(i,k)]$

Affinity Propagation groups similar code embeddings together in order to generate microservices candidates. The proposed microservices candidates are analysed using cohesion metrics in order to be compared with the results of other decomposition methods.

3.5. Metrics for Evaluating Clustering Method Performance. Silhouette coefficient, precision, recall, and f-measure were used to determine the efficiency and the threshold of the clustering method parameters. Silhouette coefficient $s(i)$ [28] is a validation method for clustered data. It measures the similarity of an object within its cluster and compares it to other clusters. An object is perfectly matched to its cluster when it gets a score of 1, and it is incorrectly matched when it gets a score of -1, so $s(i)$ score ranges from -1 to 1. The silhouette coefficient was used to evaluate the effectiveness of the clustering method with different parameters setup. The silhouette coefficient is shown in equation 3.4.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.4}$$

where $a(i)$ is the mean dissimilarity for object i, compared to the other objects in the same cluster. $b(i)$ is the smallest average distance between i and other data points in different clusters. Also, a grid search was utilized in order to find the most suitable values for the cluster algorithm parameters.

Besides the silhouette coefficient, precision and recall [29] were used to measure the performance of the clustering algorithm and its parameters. These metrics measure the efficiency of the information retrieval

method and how the retrieved results by the method are related to the requested data. Precision is defined as shown in equation 3.5.

$$P = \frac{TP}{TP + FP} \tag{3.5}$$

where P is precision, TP represent true positive results, and FP represents false positive results. The recall definition can be found in equation 3.6.

$$R = \frac{TP}{TP + FN} \tag{3.6}$$

where FN represents false-negative results. In order to get the harmonic mean of precision and recall, we used F-Measure ($F1$) to calculate the average of the precision and recall metrics, where 1 represents the best value, and 0 is the worst. $F1$ definition can be found in equation 3.7.

$$F1 = 2 * \frac{P * R}{P + R} \tag{3.7}$$

3.6. Evaluation Metrics. For this section, we chose metrics that were used by other researchers, as well. As a result, the comparison can be suitable with other decomposition methods. These researches [12], [16], and [14] used these metrics.

The first metric is Cohesion at Message Level (CHM) which uses the average cohesion of microservices interfaces at the message level. It is a refined version of Lack of Message Level Cohesion by Athanasopoulos et al. [18]. CHM value can be calculated, as shown in equation 3.8.

$$CHM = \frac{\sum_{j=1}^{\sum_{i=1}^{K} n_i} CHM_j}{\sum_{i=1}^{K} n_i}$$
$$\text{where} \quad CHM_j = \begin{cases} \frac{\sum_{(k,m)} fsimM(Op_k, Op_m)}{|I_i| * (|I_i|-1)/2} & if\ |I_i| \neq 1 \\ 1 & if\ |I_i| = 1 \end{cases} \tag{3.8}$$
$$fsimM(Op_k, Op_m) = \frac{(\frac{|res_k \bigcap res_m|}{|res_k \bigcup res_m|} + \frac{|pas_k \bigcap pas_m|}{|pas_k \bigcup pas_m|})}{2}$$

n_i represents the number of the interfaces of a microservice i. k represents the number of microservices candidates that were generated from the monolithic application. $CHMj$ measures the cohesion of a microservice at the message level. Op_k and Op_m represent the operations that are provided by the interface "I_i" of a microservice. The similarity between the output parameters and the input parameters are calculated by the similarity function $fsimM$. The higher value of the CHM metric is the better.

The other metric is Cohesion at Domain Level (CHD), which measures the average of the interfaces' cohesion at the domain level. It is a modified version of Lack of Domain Level Cohesion by Athanasopoulos et al. [18]. The formal definition of the metric is shown in equation 3.9.

$$CHD = \frac{\sum_{j=1}^{\sum_{i=1}^{K} n_i} CHD_j}{\sum_{i=1}^{K} n_i}$$
$$\text{where} \quad CHD_j = \begin{cases} \frac{\sum_{(k,m)} fsimD(Op_k, Op_m)}{|I_i| * (|I_i|-1)/2} & if\ |I_i| \neq 1 \\ 1 & if\ |I_i| = 1 \end{cases} \tag{3.9}$$
$$fsimD(Op_k, Op_m) = \frac{|T_{Op_k} \bigcap T_{Op_m}|}{|T_{Op_k} \bigcup T_{Op_m}|}$$

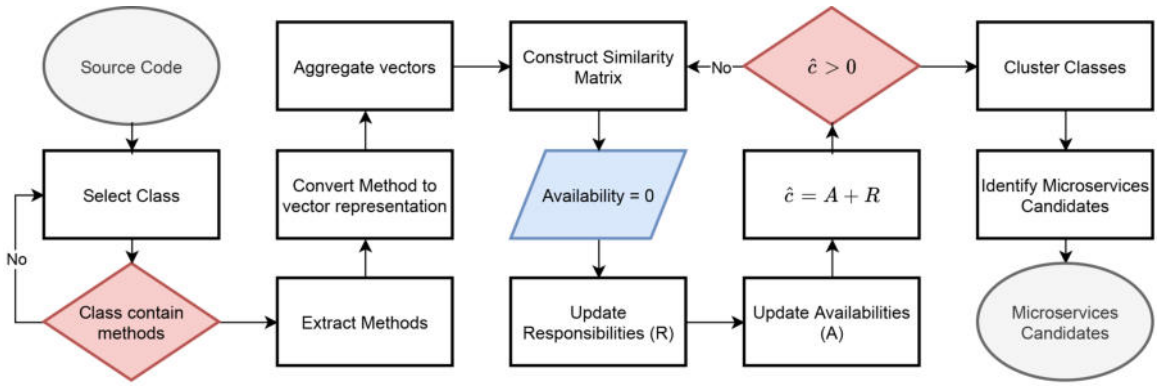


Fig. 3.4: High - Level Representation of the Proposed Algorithm

Table 4.1: Dimensions of the Tested Applications

Application	Classes	Methods	LoC	MS numbers
JPetStore	24	290	2059	4
SpringBlog	46	155	1553	6
JForum	335	2702	52,719	8
Roller	153	780	29,154	11

n_i represents the number of the interfaces of a microservice i. K represents the number of microservices candidates that were generated from the monolithic application. $fsimD$ function calculates the similarity of the operations at the domain level. Op_k and Op_m represents the domain terms that are extracted from the operations. The higher value of the CHD metric is the better.

CHM and CHD metrics were introduced by Jin et al. [12]. These metrics are used for measuring the cohesion at message and domain levels of the microservices through analysing their interfaces.

Fig. 3.4 presents a high-level description of the proposed algorithm, which starts with obtaining the methods code snippets from the monolithic application source code. Then these codes are converted to code embeddings using the code2vec model. Furthermore, aggregate the methods code embeddings using the mean function to represent the code of each class of the related methods. Finally, microservices candidates are generated through clustering related class files using a hierarchical clustering algorithm.

4. Experiments and Results. The setup of the experiment consists of testing four applications to compare the performance of the proposed method against other methods in the literature. The first application is JPetStore[1] is a pet store commercial website written in JAVA, and it is a monolithic web application consists of 24 classes. Also, it is the smallest application in the experiment setup. The second application is SpringBlog[2], which is a blogging website that is written in JAVA consists of 46 classes. The third application is JForum[3], which is a messaging boards application consisting of 335 classes. The last application is Apache Roller[4], which is a monolithic application that allows multiple users to create blog sites and posts. The sizes of these applications range from small to big applications with different class numbers, method numbers, and lines of codes. See a detailed comparison of the tested applications in Table 4.1.

[1]https://github.com/mybatis/jpetstore-6
[2]https://github.com/Raysmond/SpringBlog
[3]https://sourceforge.net/projects/jforum2/
[4]https://github.com/apache/roller

Table 4.2: Aggregation Methods Accuracy Comparison

Aggregation Method	Accuracy	Precision	Recall	F1	Silhouette coefficient
Mean	0.70	0.58	0.46	0.49	0.47
Sum	0.07	0.07	0.008	0.015	N/A
Maximum	0.15	0.33	0.10	0.14	0.27
Minimum	0.15	0.33	0.10	0.14	0.23
Median	0.23	0.56	0.27	0.30	0.17
Standard deviation	0.15	0.05	0.33	0.09	N/A
Variance	0.15	0.05	0.33	0.09	N/A

Table 4.3: JPetStore Metrics Scores

Application	Metrics	Jin et al	Our Method
JPetStore	CHD	0.52	0.52
	CHM	0.78	0.82

4.1. Aggregation Method. Several aggregation methods were tested to find the most effective method for the proposed algorithm. These methods are mean, sum, standard deviation, variance, maximum, and minimum. The setup for the experiment consisted of comparing the accuracy, precision, and recall of the clustering results against the optimal microservices design of Spring Pet Clinic[5], which have the monolithic application and the microservices design[6] as well. The results of the experiment are shown in Table 4.2. Thus, the mean function is the most suitable aggregation method for this experiment because it has the highest scores for accuracy, precision, and recall.

4.2. Clustering Method Parameters. The parameters for refining the performance of the Affinity Propagation algorithm are damping, the maximum number of iterations, and convergence iterations, the values for these parameters were 0.8, 500, and 50, respectively. These values were found using the gird search technique with different setups, configurations, and tests against the monolithic application Spring Pet Clinic which was mentioned previously. The results for these tests are displayed in Table 4.2. The tests were compared using the silhouette coefficient score.

4.3. Decomposition Results. After conducting the previous experiments and tests to find the most optimal aggregation method and the most efficient parameter values, it is the turn of displaying the results of the proposed decomposition methodology. As was mentioned before in Section 4, the decomposition method was tested with four different applications (listed in Table 4.1.)

The first application is JPetStore, which was tested by Jin et al. [12] and Saidani et al. [14]. JPetStore application was compared with Jin et al. approach in detail. For example, Fig. 4.1 shows a comparison between the decomposition results of our approach and their approach. Our approach generated four microservices while Jin et al. approach gave three microservices. Fig. 4.1 displays the microservices and their related classes.

For the cohesion side of the comparison, both of the approaches have similar results, but our approach has a slightly better score for CHM. These results in Table 4.3 are concerning the results of only JPetStore application because the decomposition results for JPetStore were described thoroughly in the research of Jin et al. [12].

The results for the comparison of the proposed method and the other methods using the additional three applications are available in Table 4.4.

[5]https://github.com/spring-projects/spring-petclinic
[6]https://github.com/spring-petclinic/spring-petclinic-microservices

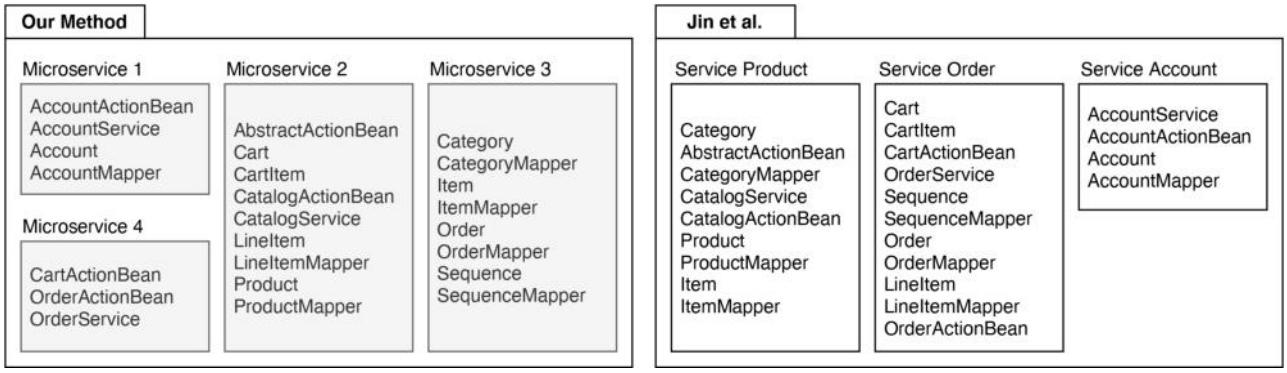


Fig. 4.1: JPetStore Results

Table 4.4: Decomposition Results

Application	Metrics	Jin et al	Saidani et al.	Our Method
JPetStore	CHD	0.52	0.65	0.52
	CHM	0.78	0.55	0.82
SpringBlog	CHD	0.55	0.67	0.50
	CHM	0.68	0.75	0.73
JForum	CHD	0.45	0.15	0.52
	CHM	0.70	0.51	0.73
Roller	CHD	0.52	0.38	0.53
	CHM	0.72	0.78	0.76

The second application is SpringBlog, which consists of 46 classes. The results in Table 4.4 suggest that our approach have a better performance in term of CHM metric compared to the other decomposition methods, but our approach has a less cohesive score, based on the CHD score, compared to the other approaches.

For the JForum application, the proposed method performed the best in terms of cohesion at the message and domain level, as it is shown in Table 4.4. It scored better scores in both CHD and CHM compared to Jin et al. and Saidani et al. methods. Therefore, this means the proposed method creates better decomposition results in terms of cohesion.

The final application is Apache Roller, where our approach had slightly improved results in term of CHD, while had a good result for CHM metrics. These results show that the proposed method can handle big applications such as JForm and Apache Roller without any issues.

The overall results for tested applications suggest that our approach has some advantages in terms of cohesion in the middle and high range applications. For example, Table 4.4 shows that most of the better and good metrics values were related to our approach, except in the small tier application such as JPetStore. Our approach scored the best results in five test experiments out of 8, while Saidani et al. method scored 3 out of 8, and Jin et al. scored 0. These results show that all the methods have good results, but the proposed method had better ones when compared with the other methods. The proposed method showed better performance in terms of cohesion, which is one of the essential requirements for a good microservices application design because microservices applications need to be loosely coupled and cohesive, according to Newman [4].

Fig. 4.2 shows an interpretation of the results that are shown in Table 4.4. Fig. 4.2 shows that our method is performing similar to Jin et al. [12] but in 4 cases has better performance. Also, the results of Saidani et al. [14] fluctuates between 0.1 and 0.8, while the results of the proposed method are between 0.5 and 0.8. Therefore, this means the proposed method has a more stable approach when compared to Saidani et al. approach.

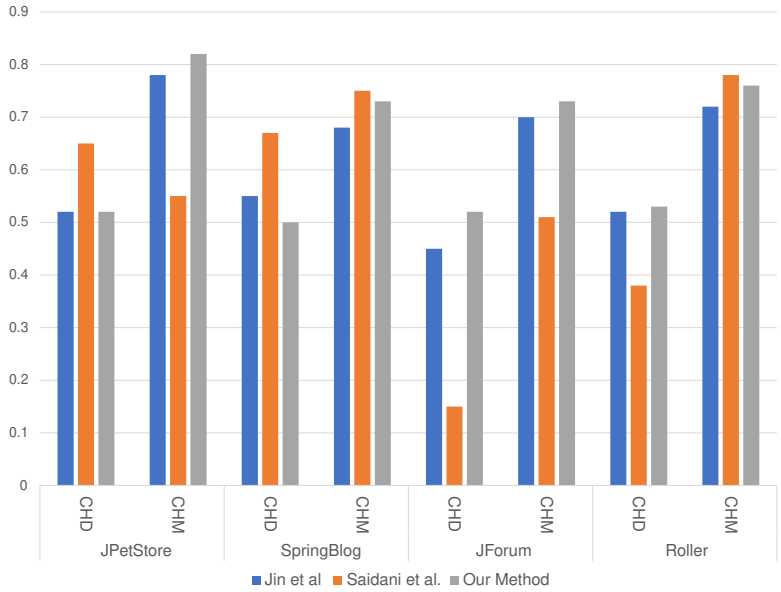


Fig. 4.2: Metrics Results Comparing the Performance of the Decomposition Methods

Table 4.5: Average Results of CHD and CHM Metrics

Metrics	Jin et al	Saidani et al.	Our Method
CHD Average	0.51	0.46	0.52
CHM Average	0.72	0.65	0.76

The overall results showed that the proposed decomposition method is better performing compared to Jin et al. and Saidani et al. methods. For example, our method had better results in 5 out of 8 metrics scores, Saidani et al. had 3, and Jin et al. 's method performed the worst when compared to the other methods. In another interpretation of the results, Table 4.5 presents the averaged results of Table 4.4, which shows that the results of Jin et al. are better on average compared to Saidani et al., but our proposed method has the best results in this case as well.

5. Conclusion. This paper proposed a novel decomposition method for refactoring monolithic applications into microservices applications using a neural network based model for creating code embeddings from the monolithic application source code. As a Result, semantically similar code embeddings are grouped using a hierarchical clustering algorithm in order to generate microservices candidates. The quality characteristics of the results were measured using two metrics for measuring cohesion.

The proposed method showed promising results in terms of cohesion when compared to other decomposition methods. The results were compared with two other methods proposed by Jin et al. [12] and Saidani et al. [14], 8 test cases were conducted, and the proposed method got the highest scores in 5 of them.

In conclusion, the proposed method can be a helpful add-on for developers in the process of migration from monolithic architecture into microservices architecture. This method will give the developers insights and directions on the path and the design that the developers need to take in order to achieve a good microservices design.

For future work, this method can be developed further and can be tested with other programming languages such as Python, C, C++, et al. The tested cases of this research were all written in JAVA, and the proposed method is only capable of handling code written in that programming language. Also, the neural network-based

model can be trained on the source codes of the microservices application to achieve more precise results.

REFERENCES

[1] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices architecture enables DevOps: Migration to a cloud-native architecture. IEEE Software 33(3):42–52, 2016.

[2] Cerny, Tomas and Donahoo, Michael J. and Trnka, Michal. Contextual Understanding of Microservice Architecture: Current and Future Directions. ACM SIGAPP Applied Computing Review 17(4):29–45, 2018.

[3] Dragoni, Nicola and Giallorenzo, Saverio and Lafuente, Alberto Lluch and Mazzara, Manuel and Montesi, Fabrizio and Mustafin, Ruslan and Safina, Larisa. Microservices: yesterday, today, and tomorrow In: Mazzara M., Meyer B. (eds) Present and Ulterior Software Engineering. Springer, Cham, 2017.

[4] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Incorporated. 2021.

[5] Nuno Santos and António Rito Silva. A complexity metric for microservices architecture migration. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 169–178, 2020.

[6] Shanshan Li, He Zhang, Z. Jia, Z. Li, C. Zhang, J. Li, Q. Gao, Jidong Ge, and Zhihao Shan. A dataflow-driven approach to identifying microservices from monolithic applications. Journal of Systems and Software 157, 2019.

[7] Omar Al-Debagy and Peter Martinek. A new decomposition method for designing microservices. Periodica Polytechnica Electrical Engineering and Computer Science 63(4):274–281. 2019.

[8] Manabu Kamimura, Keisuke Yano, Tomomi Hatano, and Akihiko Matsuo. Extracting candidates of microservices from monolithic application code. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 571–580, 2018.

[9] Genc Mazlami, Jürgen Cito, and Philipp Leitner. Extraction of microservices from monolithic software architectures. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 524–531, 2017.

[10] Luís Nunes, Nuno Santos, and António Rito Silva. From a monolith to a microservices architecture: An approach based on transactional contexts. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11681 LNCS, pages 37–52. Springer Verlag, 2019.

[11] Davide Taibi and Kari Systä. From monolithic systems to microservices: A decomposition framework based on process mining:. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science*, pages 153–164. SCITEPRESS - Science and Technology Publications, 2019.

[12] Wuxia Jin, Ting Liu, Qinghua Zheng, Di Cui, and Yuanfang Cai. Functionality-oriented microservice extraction based on execution trace clustering. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 211–218, 2018.

[13] Luciano Baresi, Martin Garriga, and Alan De Renzis. Microservices identification through interface analysis. In Flavio De Paoli, Stefan Schulte, and Einar Broch Johnsen, editors, *Service-Oriented and Cloud Computing*, volume 10465, pages 19–33. Springer International Publishing, 2017.

[14] Islem Saidani, Ali Ouni, Mohamed Wiem Mkaouer, and Aymen Saied. Towards automated microservices extraction using muti-objective evolutionary search. In Sami Yangui, Ismael Bouassida Rodriguez, Khalil Drira, and Zahir Tari, editors, *Service-Oriented Computing*, volume 11895, pages 58–63. Springer International Publishing, 2019.

[15] Muhammad Abdullah, Waheed Iqbal, and Abdelkarim Erradi. Unsupervised learning approach for web application auto-decomposition into microservices. Journal of Systems and Software 151:243–257, 2019.

[16] Wuxia Jin, Ting Liu, Yuanfang Cai, Rick Kazman, Ran Mo, and Qinghua Zheng. Service candidate identification from monolithic systems based on execution traces. IEEE Transactions on Software Engineering, 2019.

[17] Michael Gysel, Lukas Kölbener, Wolfgang Giersche, and Olaf Zimmermann. Service cutter: A systematic approach to service decomposition. In Marco Aiello, Einar Broch Johnsen, Schahram Dustdar, and Ilche Georgievski, editors, *Service-Oriented and Cloud Computing*, volume 9846, pages 185–200. Springer International Publishing, 2016.

[18] Dionysis Athanasopoulos, Apostolos V. Zarras, George Miskos, Valerie Issarny, and Panos Vassiliadis. Cohesion-driven decomposition of service interfaces without access to source code. IEEE Transactions on Services Computing 8(4):550–562, 2015.

[19] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. Science 315(5814): 972–976, 2007.

[20] Marc Mézard. Computer science. Where are the exemplars? Science 315(5814): 949–951, 2007.

[21] R. Refianti, A. B. Mutiara, and A. A. Syamsudduha. Performance evaluation of affinity propagation approaches on data clustering. International Journal of Advanced Computer Science and Applications 7(3), 2016

[22] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. code2vec: learning distributed representations of code. Proceedings of the ACM on Programming Languages 3:1–29, 2019.

[23] Brahmaleen Kaur Sidhu, Kawaljeet Singh, and Neeraj Sharma. A machine learning approach to software model refactoring. International Journal of Computers and Applications 0(0):1–12, 2020

[24] Boukhdhir Amal, Marouane Kessentini, Slim Bechikh, Josselin Dea, and Lamjed Ben Said. On the use of machine learning and search-based software engineering for ill-defined fitness function: A case study on software refactoring. In Claire Le Goues and Shin Yoo, editors, *Search-Based Software Engineering*, Lecture Notes in Computer Science, pages 31–45. Springer International Publishing, 2014.

[25] Yasemin Kosker, Burak Turhan, and Ayse Bener. An expert system for determining candidate software classes for refactoring. Expert Systems with Applications 36(6): 10000–10003, 2009.

[26] Rhys Compton, Eibe Frank, Panos Patros, and Abigail Koay. Embedding java classes with code2vec: Improvements from variable obfuscation. In Proceedings of the 17th International Conference on Mining Software Repositories, 243-253, 2020.

[27] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. A general path-based representation for predicting program properties. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 404–419, 2018.

[28] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20:53–65, 1987.

[29] Kai Ming Ting. Precision and recall. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 781–781. Springer US, 2010.

Edited by: Dana Petcu
Received: Nov 18, 2020
Accepted: Jan 21, 2021


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 53–66, DOI 10.12694:/scpe.v22i1.1839

A NOVEL SENTIMENT ANALYSIS FOR AMAZON DATA WITH TSA BASED FEATURE SELECTION

ANAND JOSEPH DANIEL D.* AND JANAKI MEENA M.†

Abstract. Sentiment analysis of online product reviews has become a mainstream way for businesses on e-commerce platforms to promote their products and improve user satisfaction. Hence, it is necessary to construct an automatic sentiment analyser for automatic identification of sentiment polarity of the online product reviews. Traditional lexicon-based approaches used for sentiment analysis suffered from several accuracy issues while machine learning techniques require labelled training data. This paper introduces a hybrid sentiment analysis framework to bond the gap between both machine learning and lexicon-based approaches. A novel tunicate swarm algorithm (TSA) based feature reduction is integrated with the proposed hybrid method to solve the scalability issue that arises due to a large feature set. It reduces the feature set size to 43% without changing the accuracy (93%). Besides, it improves the scalability, reduces the computation time and enhances the overall performance of the proposed framework. From experimental analysis, it can be observed that TSA outperforms existing feature selection techniques such as particle swarm optimization and genetic algorithm. Moreover, the proposed approach is analysed with performance metrics such as recall, precision, F1-score, feature size and computation time.

Key words: Tunicate Swarm Algorithm, Feature optimization, sentiment analysis, classifier, machine learning.

AMS subject classifications. 68T05

1. Introduction. The enormous growth of social networks including blogs, forum discussions and e-commerce sites are used among people to share their opinion about online products, services or any topics. These online reviews can be used by many organizations to enhance customer satisfaction, to improve product quality and identify the aspects of products to be upgraded. It also helps to make better-informed decisions towards user interest and preference thereby generating more profits. For example, Amazon collects customer reviews about the products or services. Similarly, social networks like Twitter and Facebook allow their users to share their opinions on any topic like events, elections, products or services. These opinions are useful to manufacturers as well as customers [1]. The manufacturers collect the negative reviews from the customers and rectify them to increase the sales. This kind of analysis is usually called as sentiment analysis (SA) termed as the computational opinion study for several events, topics, products, entities and their qualities. Hence, it is necessary to construct a sentiment analyser for the classification of product data into negative or positive sentiments and for automatic identification of product aspect sentiments from the review documents.

Based on the mechanisms used, SA is mainly classified under three forms like machine learning (ML), lexicon-based and hybrid [2]. In ML-based approaches, various learning algorithms and labelled datasets are utilized to train the classifier for identifying the sentiments [3]. In lexicon-based approaches, the sentiment polarity of the dataset is calculated through the semantic orientation of words [4]. It will classify the texts using unlabelled training set where the lexicons are defined independently of the text; so that overfitting at any instance can be prevented [28]. It showed robust performance across texts and domains and can be applied to perform SA on multiple domain datasets [6]. Though this approach posses more merits, the need for manual maintenance and less accuracy becomes a major disadvantage. Moreover, it could not identify abbreviations in a non-standard form which are mainly used in posts due to limited coverage on informal texts [7]. On the other hand, the ML approach is very suitable for informal text and unstructured content. It provides

*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, India (danny02.20099@gmail.com).

†School of Computing Science and Engineering, Vellore Institute of Technology, Chennai Campus, India (janakimeena.m@vit.ac.in).

more flexibility and hence eliminates predefined lexicons. Although ML approach performs well, it requires a manually annotated training dataset [8]. As a result, hybrid methods are introduced to bridge the gap between ML and lexicon-based approaches [9]. In this paper, valence aware dictionary for sentiment reasoning (VADER) is utilized for lexicon-based approaches and linear support vector machine (LSVM) classifier is utilized for ML approaches.

Since high-quality word embeddings can be obtained from Glove [10] and fastText [11] models, the proposed work utilizes these models to get initial word embeddings. However, including all the possible features will grow the feature size that would not fit in the memory and cause scalability problem. An approach with better scalability will be able to maintain its performance even with larger datasets. This is usually achieved by adopting suitable feature selection methods [12]. The main aim of feature selection is to generate a well-selected subset of feature that can improve the performance of classifier with better scalability and minimize the time cost simultaneously. Conversely, feature selection through optimization algorithms can automatically select features without manual intervention and eliminate large quantity of unnecessary features without compromising the accuracy.

Conventional feature selection techniques based on concepts like data pick up, shared data and Chi-square are better in reducing the redundant features but have less accuracy. Furthermore, the unwanted features create a non-polynomial hard issue which reduces the system efficiency on selection [13]. Therefore, the attention is now been shifted to intelligent algorithms to enhance classification and to solve the scalability issues. These intelligent algorithms are nature-inspired algorithms that are effective for solving complex problems during classification [14]. It reduces any kind of problems related to accuracy, rate of misclassification and error. Particle swarm optimization (PSO) [15] and genetic algorithm (GA) [16] are two traditional algorithms employed to resolve many complex problems. PSO is influenced by two notable drawbacks such as diversity loss and outdated memory [17]. This can be overcome with a novel nature-inspired algorithm called tunicate swarm algorithm (TSA) [18]. It is effective with redundant data and selects the appropriate features within least execution time.

The main aim of this paper is to propose a hybrid SA method by combining lexicon-based and ML approaches. The data is initially labelled by using VADER sentiment lexicon and the labelled data input to ML classifier. Feature selection is carried out with novel tunicate swarm algorithm. Experiments are performed to show the performance of proposed method with Amazon product reviews from four categories such as Electronics, Toys, Furniture and Camera. The performance is evaluated in terms of metrics such as F1 score, recall, precision and accuracy. Moreover, the computation time comparison of proposed TSA based feature selection with existing methods like GA and PSO is also provided. The experimental results of the proposed method with Amazon dataset shows best classification performance in each test data while adapting more training data. The rest of this paper is arranged as follows. Section 2 details the related works in the area of hybrid SA and optimized feature selection. Section 3 explains the proposed methodology. Experiments are provided in Sect. 4. In Sect. 5, the results of experiments are discussed. Section 6 concludes this paper with future work.

2. Related Works. In this section, the related works carried out in the area of lexicon-based approach, ML based approach and hybrid approach with feature selection are discussed.

2.1. Hybrid Sentiment Analysis. The lexicon-based sentiment analysis methods are widely classified into two approaches: (i) dictionary based and (ii) corpus based [19]. In dictionary-based approach, the words with their polarity scores are utilized for sentiment analysis. In corpus-based approach, positive and negative set of words together with their probability of sentiment is utilized. VADER is a lexicon as well as rule-based SA framework that performs equally or better when compared to existing SA lexicons [20]. Anton and Martin [21] applied the VADER sentiment lexicon with support vector machine (SVM) to classify the sentiments of customer response. Their method is effective with mean AUC of 0.896 and F1 score of 0.834. Tanjim et al. [22] proposed a supervised learning method for SA. Five various ML methods are utilized by them to train the classifier for Amazon product reviews and attained classification accuracy up to 94.02%. When comparing different ML classifiers, LSVM shows the best performance closely followed by random forest (RF). A similar comparison is presented in the proposed work for classifier performance evaluation and accurate measurement.

In lexicon-based methods, the polarities and frequencies of the negative and positive words are examined to get the sentiment of analysed text using a predefined dictionary of words [23]. In ML based approach,

features are generated from the text and used by different learning algorithms to predict a label. A hybrid approach is necessary to eliminate the disadvantages and to combine the merits of each methodology. The hybrid classifier is designed by using ML and lexicon-based methods to classify the documents and to improve the sentiment classification. Term polarities from lexicons are utilized as extra features to train ML classifiers in hybrid approaches [23]. Combination of these methods helps to improve the result of classifier. Xia. et al. [24] presented a hybrid approach with the combination of both ML and lexicon-based methods for SA. Kang et al. [25] introduced a combination of naive bayes (NB) and lexicon-based method for SA of reviews of restaurant. Govindrajan [26] implemented classification-based hybrid SA approach with arcing classifier. The performance of both NB and GA ensemble classifiers are analysed in terms of accuracy. Geetika et al. [27] presented set of approaches to ML with lexical analysis for the product reviews and sentence classification.

A hybrid SA technique on Facebook to automatically analyse sentiments of online product reviews has been proposed by Ortigosa et al. [28]. Zou et al. [29] introduced a method for finding imbalanced threshold classification. Agrawal and Nandi [30] proposed a layered hybrid method for SA. It has two layers where the primary layer is based on the lexicon-based method and the secondary layer is based on the ML method. Rajganesh et al. [31] introduced a hybrid method for SA of feedback-based recommendation system. A hybrid method has been implemented for SA of product reviews from Amazon [32] . In [33], a hybrid SA approach has been introduced by integrating both ML and polarity-based lexicon methods. But these existing works did not employ any optimization algorithms in the feature selection task to improve the scalability and accuracy of hybrid SA.

2.2. Optimized Feature Selection in SA. Feature selection in sentiment analysis plays a vital role in system accuracy enhancement. Many studies have been done in text classification domain with optimized feature selection. Bio-inspired algorithms have an incredible ability in solving different optimization problems [34]. Kalarani et al. [35] proposed the firefly algorithm (FA) for feature reduction with SVM and ANN classifier to classify the sentiments of movie reviews. The experimental outcomes displayed an improved accuracy with reduced training time. Kristiyanti et al. [36] implemented three algorithms called principal component analysis (PCA), PSO and GA based feature reduction with SVM classifier. The accuracy of SVM classifier has been enhanced with these three algorithms. They used Amazon products' reviews for classification. When compared with GA and PCA, PSO algorithm showed higher accuracy for classification.

Farkhund et al. [1] introduced a novel GA based feature selection technique to enhance the scalability issue that happens when the feature-set size increases in the SA. A hybrid SA approach with reviews from IMDB, Yelp and Amazon is utilized. The performance is analysed in terms of recall, accuracy, precision, F1 score and six different classifier algorithms. The results proved that GA based feature selection approach obtains higher accuracy than LSA (Latent semantic analysis) and PCA methods. In the proposed work, a hybrid framework is utilized for SA with novel TSA based optimal feature selection for testing ML classifiers. In sentiment classification, PSO with SVM classifier is the most commonly used approach among researchers followed by ant colony optimization (ACO) [34]. But PSO based optimization has some disadvantages such as the need for multi-objective optimization, fine-tuning of parameters value and poor performance for datasets of multi-domain [37].

This paper proposes a novel TSA based feature reduction in SA. The proposed feature selection approach is encouraged by swarm and jet propulsion behaviours of tunicates throughout the foraging and navigation process [18]. TSA can reduce local optima problem and deliver fast convergence speed by providing better performance in exploration and exploitation stages. Furthermore, it has the capability to keep the stability between exploitation and exploration by searching the large space to get the best global solution. Thus, TSA will be a better solution for feature selection tasks to overcome

3. Proposed Methodology. This section explains the hybrid SA with TSA based optimized feature selection approach. The proposed methodology with VADER sentiment lexicon and ML classifiers is detailed. Then TSA based optimized feature selection is explained

3.1. Hybrid SA with Optimized Feature Selection Approach. This hybrid method includes lexicon-based dictionary for training the ML classifier and bag-of-words as features for testing the ML classifier with TSA based feature selection. A complete framework of proposed work is displayed in Fig. 3.1.

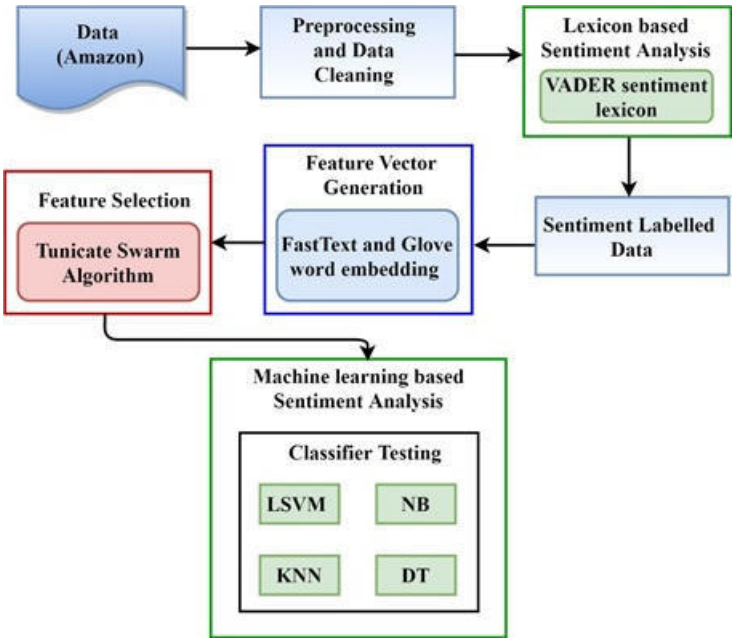


Fig. 3.1: Block diagram of proposed hybrid SA methodology

In the proposed method, the Amazon product data is utilized for SA. Initially, the dataset is pre-processed to eliminate unnecessary data. VADER sentiment lexicon is performed to label the pre-processed data. The feature vectors are generated for labelled training data using Glove and fastText word embeddings. If all the feature vectors are directed to the ML classifier, scalability issue may arise because these vectors contain 80% of the input data. As the dataset size grows bigger, this problem worsens. To minimize this problem, an efficient TSA based feature selection process is introduced in this paper. In this method, each feature vector is modelled on TSA for several hundred generations. TSA simulation is run to find the optimal feature vectors to give improved accuracy for SA. These selected features are utilized to train the ML based classifiers. Linear Support Vector machine, Decision Tree (DT), Naïve Bayes and k-Nearest Neighbours (KNN) are the ML classifiers utilized for classification in this proposed work. The description of these classifiers is given below. Manually labeled dataset is utilized to validate the performance of proposed hybrid SA model.

LSVM. SVM is one of the most used supervised learning techniques which can perform regression, and classification. However, SVM is widely utilized for classification in ML. SVM can handle simple and complex datasets with higher accuracy than other algorithms. In classification, SVM transforms the data points and find hyperplane with maximum margin from multiple decision boundaries to classify the data points in n-dimensional space using the kernel trick technique. The polynomial, linear or Gaussian RBF kernel can be utilized to minimize the computational complexity related to the prediction of new data point. Training an SVM with a linear kernel is quicker than with any other kernel. NB: It is also a supervised learning technique mainly utilized in text classification. NB classifiers are based on the Bayes' theorem and occurrence of a particular feature is independent of occurrence of other features. It is a probabilistic based machine learning model that predicts based on the probability of each feature. NB is capable of handle high dimensional complex datasets.

KNN. It is also one of the simplest supervised learning techniques that is utilized for both regression and classification. To classify new data this algorithm assumes the similarity between training data and new data. Then classifies the data based on the similarity. The data is assigned to the category for which the similarity is maximum. To assume the similarity, the distance between training data and testing data is measured by using Euclidean distance. The number of nearest neighbours is calculated for test data. The category of new data is

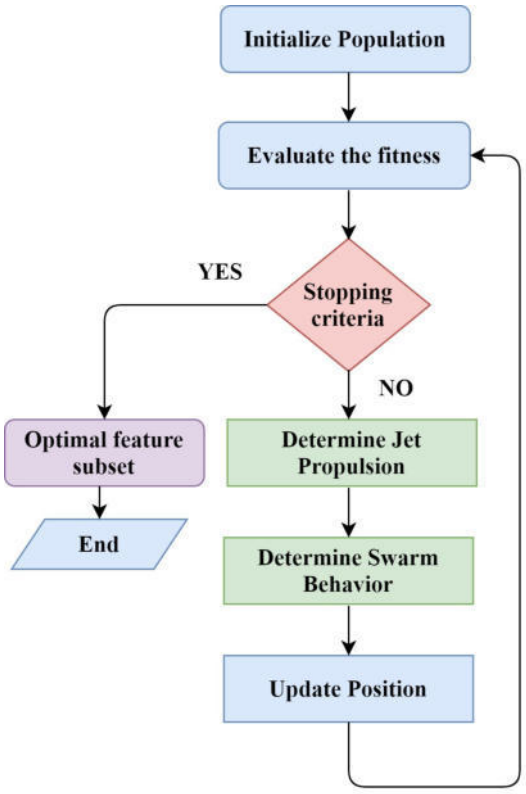


Fig. 3.2: Block diagram of proposed hybrid SA methodology

selected for which data have a greater number of nearest neighbours.

DT. It is also a supervised learning technique that can be utilized to perform both regression and classification. DT utilizes tree structure to classify the data based on given conditions in which root node represents the whole training dataset, decision rules such as Boolean function are represented as branches and output class label is represented by each leaf node. The DT algorithm starts with root node which contains whole dataset. The best attributes are selected using the Attribute Selection Measure. The decision node is created with best attributes. This process is repeated until finding the leaf node for all branches.

3.2. Feature Selection using TSA. When using a larger dataset, scalability issue arises as the increase in the feature vector size. This issue can be eliminated with feature vector optimization by minimizing its size when keeping the accuracy without reduction. This problem is framed in this section and its solution is proposed by utilizing TSA approach.

3.2.1. Problem formulation. If we include all the possible features in the testing set, then the increase in the size of datasets will create larger feature size that creates scalability problem. Thus, to eliminate this problem, the feature selection optimization technique is required. An optimization problem is the minimum number of selection of features from the feature-set of large size. As discussed before, the TSA has more merits such as achieving optimized fitness value and early convergence compared to other optimization algorithms like PSO and GA. Also, this optimization algorithm does not need labelled dataset for sentiment classification.

3.2.2. Mathematical model and optimization. From the training data, the feature vectors are extracted and the optimal feature subset is selected by using TSA approach. Fig. 3.2 shows the flow chart of the proposed TSA based feature selection approach.

In the proposed approach, the optimal feature subset is selected according to the position of best features

($\overrightarrow{FS}$). The features are initialized as the tunicate population ($\overrightarrow{P_p}$). The features can keep its position towards the best feature. The best feature is explored after the computation of fitness value of each feature. In the proposed approach, the feature subset having minimum error rate in prediction of sentiment polarity is considered as the best feature. The fitness value is calculated using Eq. 3.1.

$$fitness = min(1 - accuracy) \tag{3.1}$$

Here, accuracy in the prediction of sentiment polarity is considered to calculate the error rate. Algorithm 3.1 explains the fitness function calculation.

Algorithm 1: Fitness calculation

Procedure *Compute Fitness* ($\overrightarrow{P_p}$)
  for $i \leftarrow 1toN$ **do**
    Fit [i] = FitnessFunction $(P_p)(\overrightarrow{i}, :))$
  $Fit_{best} \leftarrow Best(Fit[])$;
  return Fit_{best}
Procedure *Procedure Best (Fit)*
  $Best \leftarrow Fit[0]$;
  for $i \leftarrow 1toN$ **do**
    if $(Fit[i] < Best)$ **then**
      $Best \leftarrow Fit[i]$
  return *Best*

The position of best feature $\overrightarrow{(FS)}$ is explored after the computation of fitness value. $\overrightarrow{(PD)}$ is defined as the distance between the features and the position of best feature. It can be determined using Eq. 3.2.

$$\overrightarrow{(PD)} = |\overrightarrow{(FS)} - r_{and}.\overrightarrow{(P_p(x))}| \tag{3.2}$$

where x represents the present iteration, (r_{and}) represents a random number in the range between 0 and 1. ($\overrightarrow{A}$) is calculated using Eqs. 3.3, 3.4 and 3.5.

$$\overrightarrow{A} = \frac{\overrightarrow{G}}{\overrightarrow{M}} \tag{3.3}$$

$$\overrightarrow{G} = c_2 + c_3 - \overrightarrow{F} \tag{3.4}$$

$$\overrightarrow{F} = 2.c_1 \tag{3.5}$$

Here $\overrightarrow{G}$ represents the force of gravity and $\overrightarrow{F}$ represents water flow advection in deep ocean [17]. $c_1, c_2 and c_3$ are random number variables between the range [0, 1]. $\overrightarrow{M}$ shows the tunicates social forces. $\overrightarrow{M}$ is calculated as shown in Eq. 3.6.

$$\overrightarrow{M} = \lfloor P_{min} + c_1.P_{max} - P_{min} \rfloor \tag{3.6}$$

where P_{min} and P_{max} shows the initial and secondary speeds to create social interaction. P_{min} and P_{max} values are considered as 1 and 4 respectively. $\overrightarrow{(P_p(x))}$ is the position of feature which is calculated using Eq. 3.7.

$$\overrightarrow{P_p(x)} = \begin{cases} \overrightarrow{FS} + \overrightarrow{A}.\overrightarrow{PD}, & \text{if } r_{and} \geq 0.5 \\ \overrightarrow{FS} - \overrightarrow{A}.\overrightarrow{PD}, & \text{if } r_{and} < 0.5 \end{cases} \tag{3.7}$$

The position of each features $\overrightarrow{P_p(x)}$ is updated according to the position of best feature $\overrightarrow{FS}$ using Eq. 3.8.

$$\overrightarrow{P_p(x+1)} = \frac{\overrightarrow{P_p(x)} + \overrightarrow{P_p(x+1)}}{2 + c_1} \tag{3.8}$$

The updated features going beyond the optimal feature size are adjusted. The fitness value is calculated for the updated position of features. ($\overrightarrow{P_p}$) is updated till a solution is found better than the prior optimal solution. This process is repeated until the satisfied stopping criterion is obtained. Thus, the best optimal feature subset is obtained from the OBL-TSA based feature selection process. Algorithm 3.2 explains the optimal feature subset selection with OBL-TSA based feature selection process.

Algorithm 2: Feature selection with Tunicate Swarm Algorithm

input : Feature set as tunicate population $\overrightarrow{P_p}$, N dimension of features.
output: Position of optimal best features $\overrightarrow{FS}$
Procedure *Procedure TSA_feature_selection*
  //Initialization;
  Initialize population $\overrightarrow{P_p}$;
  Initialize the parameters $\overrightarrow{A}$, $\overrightarrow{G}$, $\overrightarrow{F}$, $\overrightarrow{M}$, and Max_{iter};
  Set $\overrightarrow{P_p} \leftarrow 0$, $P_{min} \leftarrow 1$, $P_{max} \leftarrow 4$;
  while $x < Max_{iter}$ **do**
    //Compute fitness;
    Calculate position of best features $\overrightarrow{FS}$ using Eq. 3.1;
    $\overrightarrow{FS} \leftarrow ComputeFitness(\overrightarrow{P_p})$ using;
    // Jet propulsion and swarm behaviour;
    $c_1, c_2, c_3, r_{and} \leftarrow Rand()$;
    Determine $\overrightarrow{PD}$ using Eq. 3.2,3.3,3.4,3.5 and 3.6.;
    Calculate the position of features $\overrightarrow{P_p(x)}$ using Eq. 3.7;
    Update the position of features using Eq. 3.8;
  return $\overrightarrow{FS}$

4. Experiments. This section details which dataset used in this work, how dataset is prepared for SA by removing the redundant data and also explains how to evaluate the proposed model for SA tasks.

4.1. Dataset. The dataset utilized for this experiment is Amazon product reviews from four categories such as Electronics, Toys, Furniture and Camera. It consists of approximately 48,500 product reviews extracted from Amazon.com ($https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt$). The dataset after manual data cleaning process has been taken for analysis. This contains 1000 positive reviews and 1000 negative reviews for each category. Each review has information with rating (0–5 stars), reviewer location and name, review date and title, product name and the review text. The dataset was unlabelled and labelled manually while using as testing data. For that purpose, the ratings of product reviews have been divided into two categories, reviews with ratings >3 labelled as positive and the reviews with rating <3 labelled as negative. These reviews go through data cleaning and pre-processing before given as input to ML classifier because of its unstructured format.

4.2. Data Pre-Processing and Cleaning. In this section, product reviews after manual processing are kept in the memory for pre-processing and cleaning. Pre-processing of data removes unwanted data including web addresses, URLs and online links. This module also includes tokenization and case conversion.

- Removal of unwanted data. This step removes unwanted non-characters consisting of URLs, symbols, web addresses, digits and online links from the review.

Table 5.1: Model Parameters

Algorithm	Parameters	Values
PSO	Weight	0.2
	Constant	2
GA	Rate of Crossover	0.8
	Rate of Mutation	0.01
Proposed TSA	P_{min}	1
	P_{max}	4

- Removal of stop words. Most of the more regularly used stop words in English are “an”, “of”, “a”, “you”, “the”, “and”, etc. These are some words that do not have any meaning. So, these words are generally ignored to improve the accuracy of SA. These words are collected together and removed from the dataset.
- Tokenization. In this process, the sequence of strings is separated into individuals such as keywords, words, symbols, phrases and tokens based on the space of separation. Further, punctuation marks are discarded in this process.
- Case conversion. All the reviews should be converted to lower case because it must be in same case to process. At last, a string of meaningful words is obtained.

4.3. Performance Metrics. In this paper, four performance measures were utilized for the performance evaluation of proposed approach: precision, accuracy, recall and F1 score. The performance measures are given as follows.

- True Positives. It is the count of positive comments classified correctly.
- False Positives. It is the count of negative comments classified wrongly.
- True Negatives. It is the count of negative comments classified correctly.
- False Negatives. It is the count of positive comments classified wrongly.
- Accuracy. It is the ratio between the correctly classified comments and the total number of comments as shown in Eq. 4.1.

$$accuracy = TP + TN/TP + TN + FP + FN \tag{4.1}$$

- Precision. It is the ratio of properly classified positive comments over the total number of positive classified comments as shown in Eq. 4.2.

$$precision = TP/TP + FP \tag{4.2}$$

- Recall. It is the ratio of properly classified positive comments over all comments actually belonging to that class as shown in Eq. 4.3.

$$recall = TP/TP + FN \tag{4.3}$$

- F1-Score. It is the weighted average of recall and precision as shown in Eq. 4.4.

$$F1 = 2 * precision * recall/precision + recall \tag{4.4}$$

5. Parameter Settings. The experiments are performed on a HP laptop with Windows 10 operating system, Intel Core i3 processor having 2.3 GHz frequency, 4GB of RAM. The software used for evaluation of the proposed framework is MATLAB R2020a. The model parameters used to validate the performance of various optimization techniques in SA oriented feature selection domain and general feature selection domain is given in Table 5.1.

Table 6.1: Performance measures comparison of different classifiers in sentiment analysis

Classifier	Accuracy	Precision	Recall	F1 score	Computation Time (sec)
LSVM	93	94.3	97.6	96	22
NB	85	89.7	94.3	92	28
DT	82	88.8	88.4	88.6	32
KNN	87	92.9	96.5	94.7	25

Table 6.2: Performance measures comparison of VADER lexicon model

Products	Accuracy	Precision	Recall	F1 score
Furniture	68	71.3	65.3	68.3
Toys	62	63.5	56.8	59.9
Electronics	63	65.3	57.7	61.5
Camera	59	61	58.4	59.7

6. Experiment Results and Discussion. The experiment results and discussion are given in this section. The performance of the proposed SA approach is evaluated using four different ML based classifiers: LSVM, NB, DT and KNN. The accuracy, recall, precision and F1 score are used as performance measures for classifiers.

6.1. Performance Comparison of ML classifiers. The comparison of four various ML classifiers in the proposed hybrid SA approach is performed by applying TSA and non-TSA enhanced features. The same tests are performed for the reviews of four different products that have been discussed previously.

Table 6.1 shows the accuracy, precision, recall and F1 score comparison for four different classifiers. These results are taken for SA of Furniture dataset. The accuracy of LSVM is higher than other classifiers. It has achieved more than 90% accuracy which lies below 90% for remaining classifiers. The precision rate of all the classifiers are observed to be more than 85%. However, precision rate of LSVM is comparatively higher than other classifiers. The recall measure and F1-score of all classifiers except DT attained more than 90%. The recall rate is 1% higher for LSVM compared to KNN approach. Likewise, F1-score is slightly better for LSVM in comparison to both NB and KNN. When considering computation time, LSVM takes less time than other classifiers. From these results, it can be observed that total performance of LSVM classifier is better than other classifiers. In the proposed work, LSVM classifier is used for classification of other datasets since it proves best accuracy than other classifiers.

6.2. Comparison of three different SA approaches. Three SA approaches are implemented in the proposed SA framework. Table 6.2 shows the performance comparison for VADER sentiment lexicon model. The accuracy of all datasets except Camera dataset attained more than 60% . The accuracy of Furniture dataset achieved 68% which lies below 65% for other datasets. In terms of precision, the Furniture dataset has the maximum precision of 71.3%. Similarly, the F1 score of Furniture dataset is comparatively higher than other datasets. The Recall rate is 65.3% for Furniture dataset.

Table 6.3 shows the performance measures comparison of ML approach for LSVM classifier with TSA based feature selection approach. The fastText and Glove word embedding model is utilized to create feature vectors. From Table 4, it can be observed that Camera dataset has the maximum accuracy than other datasets. The precision rate of Furniture dataset is comparatively higher than other datasets. In terms of Recall, Camera dataset shows a higher recall than Electronics dataset. The F1 score of all datasets was observed to be more than 70%. The F1score is slightly better for Camera dataset in comparison to Electronics dataset.

Table 6.4 shows the results of the proposed model with TSA and without TSA. LSVM classifier is utilized to classify the Amazon product data as it shows better results. From Table 5, it can be observed that the accuracy of TSA and non- TSA based approaches for Furniture dataset achieved maximum accuracy than

Table 6.3: Performance measures comparison of ML approach with LSVM classifier

Products	Accuracy	Precision	Recall	F1 score
Furniture	79	81.8	62	71.9
Toys	74	75.2	71.6	73.4
Electronics	78	78	78.8	78.4
Camera	85	70.2	87.2	78.7.7

Table 6.4: Performance measures comparison of proposed hybrid model with TSA and without TSA on Amazon data

Products	**With TSA**				**Without TSA**			
	Accuracy	**Precision**	**Recall**	**F1 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Furniture	93	94.4	97.6	96	91	93.9	97.3	95.6
Toys	86	92.5	95.3	93.9	84	89.8	91.6	90.7
Electronics	89	93	96.2	94.6	88	89.3	95.1	92.2
Camera	85	87.5	93.1	90.3	82	85.4	92.8	89.1

other datasets. The precision rate of all datasets achieved above 90% except Camera dataset for both TSA and non- TSA optimization. The Recall and F1 score is more than 90% for all datasets with TSA optimization. Without TSA optimization, the Recall rate is above 90% for all datasets. Likewise, F1 score is above 90% for all datasets except Camera dataset without TSA optimization. These results show that the performance measures of TSA based optimized feature selection is almost similar to non-TSA based technique with reduced feature size. Thus, TSA based optimal feature selection reduces the scalability problem and computation time than non- TSA based approaches. Thus, Tables 3, 4 and 5 shows that the proposed hybrid SA approach with TSA based feature selection outperforms VADER lexicon and ML-based approaches.

6.3. Feature size comparison of TSA based ML approach. Feature size comparison is performed to show that TSA based feature selection improves the scalability by reducing the feature size while keeping the same accuracy as non-TSA based SA.

Figure 6.1 shows the feature size before and after SA optimization on selection of features. For Electronics dataset, the size of features before employing TSA is 3295. After employing TSA, the feature size has been minimized to 2169 which is nearly 34% less in the size. For the Toys dataset, the size of features before employing TSA is 4045. After employing TSA, the feature size has been minimized to 2985 which is almost 26% less in the size. For Camera dataset, the size of features before employing TSA is 2940. After employing TSA, the feature size has been minimized to 1754 which is almost 40% less in the size. Moreover, for Furniture dataset, the size of features before performing TSA is 3492. After performing TSA, the feature size has been minimized to 1982 which is nearly 43% less in the size. We have already seen that the performance of both methods is similar but optimization using TSA gives minimized size of the feature. This reflects an important effect on the system scalability. When dataset with large features is used for SA, it results in an enormous bottleneck.

6.4. Comparing of TSA with GA and PSO. The results are taken only for LSVM classifier because LSVM gives a better outcome for both TSA and non-TSA methods. To make a comparison and to verify the results of TSA, two famous existing algorithms: GA and PSO are considered.

Figure 6.2 illustrates that TSA finds the optimum solution faster than the other two optimization algorithms and it has an improved convergence speed. Thus, the proposed TSA based feature selection algorithm can keep the balance between the exploration and exploitation and escape from local optima problem.

From Table 6.5, it can be known that TSA based feature selection shows better accuracy of 93% which is

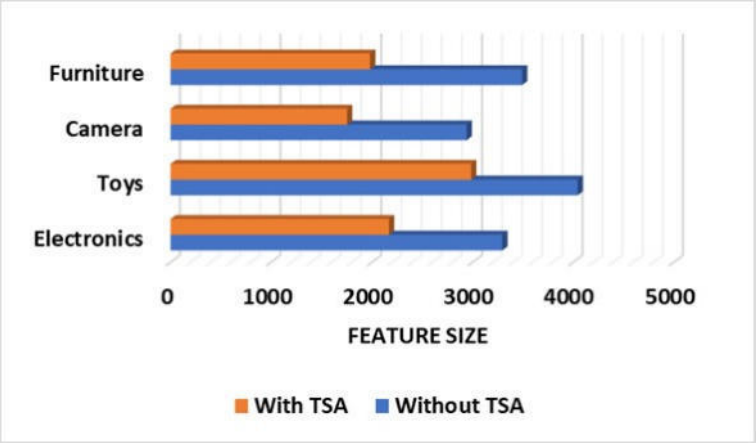


Fig. 6.1: Features size comparison for various datasets with TSA and without TSA

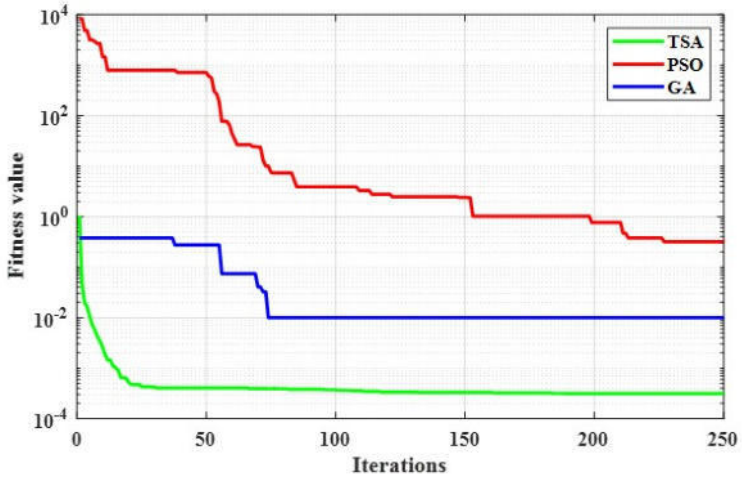


Fig. 6.2: Convergence curve comparison for PSO, GA in general feature selection domain

Table 6.5: Performance Comparison of various feature selection techniques

Techniques	**Performance measures**					
	Accuracy	**Precision**	**Recall**	**F1 score**	**Computation time**	**Feature size reduction (%)**
Proposed OBL-TSA	95	95.4	96.6	96	15 sec	43
TSA	93	94.3	97.6	96	22 sec	40
PSO	89	91.1	96.5	93.7	35 sec	38
GA	90	92.1	96.5	94.3	38 sec	35

Table 6.6: Comparative Analysis with related works

Methods	**Dataset**	**Accuracy (%)**
SA with Probabilistic Machine Learning for Amazon Reviews [38]	Books	82.9
	Electronics	86.6
	DVD	83.7
	Kitchen	89.1
SA for business analytics with and cellphone Amazon Reviews [39]	Accessories & Cellphone	80.11 72.95
SA for Amazon Product Reviews and feature selection and methods [40]	Camera	62 80 68
	Musical instruments	62 80 68
	Books	70 70 80
Proposed Model	Electronics	89
	Toys	86
	Camera	85
	Furniture	93

better than PSO and GA based feature selection with less computation time (22 sec). Moreover, 43% of the features are minimized using TSA based SA. The final part of this discussion demonstrates that TSA based feature reduction outperformed both GA and PSO based feature selection approaches. From these results, it can be concluded that the proposed TSA based approach is efficient than other commonly used existing algorithms for feature selection.

Table 6.6 shows the comparative analysis of the proposed method with the related works in terms of accuracy. The related works employed various pre-processing and feature extraction processes. In the proposed approach hybrid approach is employed with optimized feature selection. From this comparative analysis and above results, it can be observed that the proposed approach is more effective than existing optimization algorithms for feature selection and could give better results than related works.

7. Conclusion. In this paper, a novel hybrid model with feature selection using TSA is designed, developed and evaluated. Moreover, the proposed hybrid technique is evaluated for scalability in terms of execution time comparison. In the total execution time, optimal feature-set selection using TSA took about 50-60% and reduced feature-size up to 43% without any change in accuracy (93%). Experiments showed the performance of proposed method with the data of Amazon product reviews from four categories Electronics, Camera, Furniture and Toys. The evaluation of the proposed work is done with performance metrics such as F1-score, recall, precision and accuracy. The results indicated that the TSA based optimized feature selection method showed improved accuracy than PSO and GA algorithms with less computation time. Thus, the proposed approach has higher accuracy and better scalability for SA of online reviews. The future work aims at the extension of the proposed work on multi-domain SA with various sources of datasets.

REFERENCES

[1] F. Iqbal, J. Maqbool, B. C. M. Fung, R. Batool, A. M. Khattak, S. Aleem And P. C. K. Hung, *A Hybrid Framework for Sentiment Analysis using Genetic Algorithm based Feature Reduction*, IEEE Access, 1-1, 2019.

[2] S. Kausar, X. Huahu, M. Y Shabir And W. Ahmad, *A Sentiment Polarity Categorization Technique for Online Product Reviews*, IEEE Access, 8, 3594–3605, 2020.
[3] E. Boiy And M.-F. Moens, *A machine learning approach to sentiment analysis in multilingual Web texts*, Information Retrieval, 12(5), 526–558, 2008.
[4] M. Taboada, J. Brooke, M. Tofiloski, K. Voll and M. Stede, *Sentiment analysis in Facebook and its application to e-learning*, Computers in Human Behavior, 37(2), 267–307, 2014.
[5] A. Ortigosa, J.M. Martín and R.M. Carro, *Lexicon-Based Methods for Sentiment Analysis*, Computational Linguistics, 31, 527–541, 2011.
[6] G. Katz, N. Ofek and B. Shapira, *ConSent: Context-based sentiment analysis*, Knowledge-Based Systems, 84, 162–178, 2015.
[7] P. Ducange, M. Fazzolari, M. Petrocchi and M. Vecchio, *An effective Decision Support System for social media listening based on cross-source sentiment analysis models*, Engineering Applications of Artificial Intelligence,78, 71–85, 2019.
[8] Z. Yuan, S. Wu, F. Wu, J. Liu and Y. Huang, *Domain attention model for multi-domain sentiment classification*, Knowledge-Based Systems,155, 1–10, 2018.
[9] L. Shang, Z. Zhou and X. Liu, *Particle swarm optimization-based feature selection in sentiment classification*, Soft Computing, 20(10), 3821–3834, 2016.
[10] J. Pennington, R. Socher and C. Manning, *Glove: Global Vectors for Word Representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543, 2014.
[11] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, *En-riching Word Vectors with Subword Information*, Transactions of the Association for Computational Linguistics, 5, 135–146, 2017.
[12] G. Ansari, T. Ahmad and M.N. Doja, *Hybrid Filter–Wrapper Feature Selection Method for Sentiment Classification*, Arabian Journal for Science and Engineering, 2019.
[13] Y. Liu, C. Goa, Z. Zhang, Y. Lu, S. Chen, M. Liang and L. Tao, *Solving NP-Hard Problems with Physarum-Based Ant Colony System*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14(1), 108–120, 2017.
[14] A. Nabaei, M. Hamian, M.R. Parsaei, R. Safdari, T. Samad-Soltani, H. Samad-Soltani and A. Ghassemi, *Topologies and performance of intelligent algorithms: a comprehensive review*, Topologies and performance of intelligent algorithms: a comprehensive review. Artificial Intelligence Review, 49(1), 79–103, 2016.
[15] R. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory*, MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science.
[16] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*,U Michigan Press.
[17] S.S. Grill, R. Buyya, I. Chana, M. Singh and A. Abraham, *BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources*, Journal of Network and Systems Management, 26(2), 361–400, 2017.
[18] S. Kaur, L.K Awasthi, A.L. Sangal and G. Dhiman, *Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization*, Engineering Applications of Artificial Intelligence, 90, 103541, 2020.
[19] M. Darwich, S.A.M. Noah, N. Omar and N. Osman, *Corpus-Based Techniques for Sentiment Lexicon Generation: A Review*, Journal of Digital Information Management, 17, 296, 2019.
[20] C.J. Hutto and E. Gilbert, *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*, Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM, 2014.
[21] A. Borg and M. Boldt, *Using VADER Sentiment and SVM for Predicting Customer Response Sentiment*, Expert Systems with Applications, 113746, 2020.
[22] T.U. Haque, N.N. Saber and F.M. Shah, *Sentiment analysis on large scale Amazon product reviews*, IEEE International Conference on Innovative Research and Development (ICIRD), 2018.
[23] H. Cho, S. Kim, J. Lee and J.S. Lee, *Data-driven integration of multiple sentiment dictionaries for lexicon-based sentiment classification of product reviews*, Knowledge-Based Systems, 71, 61–71, 2014.
[24] R. Xia, C. Zong and S. Li, *Ensemble of feature sets and classification algorithms for sentiment classification*, Information Sciences, 181(6), 1138–1152, 2011.
[25] H. Kang, S.J. Yoo and D. Han, *Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews*, Expert Systems with Applications, 39(5), 6000–6010, 2012.
[26] M. Govindarajan, *Sentiment analysis of movie reviews using hybrid method of naive Bayes and genetic algorithm*, Int. J. Adv. Comput. Res., 3(4), 139145, 2013.
[27] G. Gautam and D. Yadav, *Sentiment analysis of twitter data using machine learning approaches and semantic analysis*, Seventh International Conference on Contemporary Computing (IC3), 2014.
[28] A. Ortigosa, J.M. Martín and R.M. Carro, *Sentiment analysis in Facebook and its application to e-learning*, Computers in Human Behavior,31, 527–541, 2014.
[29] Q. Zou, S. Xie, M. Wu and Y. Ju, *Finding the Best Classification Threshold in Imbalanced Classification*, Big Data Research,5, 2–8, 2016.
[30] V. Nandi and S. Agrawal, *Political sentiment analysis using hybrid approach*, International Research Journal of Engineering and Technology (IRJET),3(5), 1621–1627, 2016.
[31] N. Rajganesh, C. Asha, A.T. Keerthana and K. Suriya, *A hybrid feedback-based book recommendation system using sentiment analysis*, IJSRCSEIT,3(3), 2456–3307, 2018.
[32] D. Mumtaz and B. Ahuja, *A Lexical and Machine Learning-Based Hybrid System for Sentiment Analysis*, Studies in Computational Intelligence, 165–175, 2017.
[33] I. Gupta and N. Joshi, *Enhanced Twitter Sentiment Analysis Using Hybrid Approach and by Accounting Local Contextual Semantic*, Journal of Intelligent Systems, 0(0), 2019.

[34] A. Yadav and D.K. Vishwakarma, *A comparative study on bio-inspired algorithms for sentiment analysis*, Cluster Computing, 2020.
[35] P. Kalarani and S. Brunda, *Sentiment analysis by POS and joint sentiment topic features using SVM and ANN*, Soft Computing, 2018.
[36] D.A. Kristiyanti and M. Wahyudi, *Feature selection based on Genetic algorithm, particle swarm optimization and principal component analysis for opinion mining cosmetic product review*, 5th International Conference on Cyber and IT Service Management (CITSM), 2017.
[37] H. Jiang, C.K. Kwong, W.Y. Park and K.M. Yu, *A multi-objective PSO approach of mining association rules for affective design based on online customer reviews*, Journal of Engineering Design, 29(7), 381–403, 2018.
[38] R. Xia, F. Xu, J. Yu, Y. Qi and E. Cambria, *Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis*, Information Processing & Management, 52(1), 36–45, 2018.
[39] M.S. Elli and Yi-Fan, *Amazon Reviews, business analytics with sentiment analysis*,2016.
[40] T. Shaikh and D. Deepa, *Feature Selection Methods in Sentiment Analysis and Sentiment Classification of Amazon Product Reviews*,International Journal of Computer Trends and Technology, 36(4), 225-230, 2016.

Edited by: Dana Petcu
Received: Nov 26, 2020
Accepted: Jan 23, 2021


Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 67–79, DOI 10.12694:/scpe.v22i1.1840

DISTRIBUTED APPLICATION CHECKPOINTING FOR REPLICATED STATE MACHINES*

NIYAZI ÖZDINÇ ÇELIKEL† AND TOLGA OVATMAN‡

Abstract. Application checkpointing is a widely used recovery mechanism that consists of saving an application's state periodically to be used in case of a failure. In this study we investigate the utilisation of distributed checkpointing for replicated state machines. Conventionally, for replicated state machines, checkpointing information is stored in a replicated way in each of the replicas or separately in a single instance. Applying distributed checkpointing provides a means to adjust the level of fault tolerance of the checkpointing approach by giving away from recovery time. We use a local cluster and cloud environment to examine the effects of distributed checkpointing in a simple state machine example and compare the results with conventional approaches. As expected, distributed checkpointing gains from memory consumption and utilise different levels of fault tolerance while performing worse in terms of recovery time.

Key words: Application Checkpointing, Replicated State Machines, Cloud Computing

AMS subject classifications. 68M14, 68W15

1. Introduction. During the passing few years, serverless computing has become more widespread among the cloud service providers. Very broadly, this term refers to isolating almost every layer of the software development stack from service developer by providing a service modelling medium such as a state machine. By using this service definition model, developer might model and execute simple services without worrying about the configuration of software stack layers.

From a cloud provider's perspective, using replicated state machine (RSM) approach for fault tolerance is a favourable alternative [1] [2] since it is a widely-known and implemented approach among software developers, there even exists many frameworks for back-end programming such as Spring State Machines[1]. RSMs simply execute replicas of a state machine to handle requests in a distributed way. During running time, each replica handles different requests and executes them as if they are being orderly processed by a main state machine.

An example of state machine replication can be seen in Fig. 1.1, where a master state machine on top is replicated over three replicas. State machines transit between defined states such as A, B and C with incoming events such as E1, E2 and E3. Using a master replica (or state machine) is dependent on the context of usage. When no master is used, replicas are expected to eventually be orchestrated to reflect a single logical state machine. The replicas can be deployed in proximate locations as well as in geographically distinct locations [3] that may affect the performance of orchestration among the replicas.

One of the important aspects in deploying RSMs is fault recovery. Replicas may periodically save system state, known as checkpoints, to recover to a past state in the presence of a system failure. Checkpointing is also utilised for the cases where a new replica is introduced to the system to update the replica's state to the current state of the RSM. For RSMs, using different checkpointing approaches might have different characteristics in terms of non-functional properties of the system. For instance, if each replica keeps full restore information specific to the replica, redundant replicated checkpointing information would emerge since all the replicas eventually go through the same execution path at run-time. On the other hand, keeping a single checkpointing replica would result in a single point of failure for checkpointing operation.

*This work is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant id 118E887.
†Istanbul Technical University Department of Computer Engineering Istanbul, Turkey (celikelni@itu.edu.tr).
‡Istanbul Technical University Department of Computer Engineering Istanbul, Turkey (ovatman@itu.edu.tr) ORCID-id:0000-0001-5918-3145. Corresponding author.
[1]https://projects.spring.io/spring-statemachine/

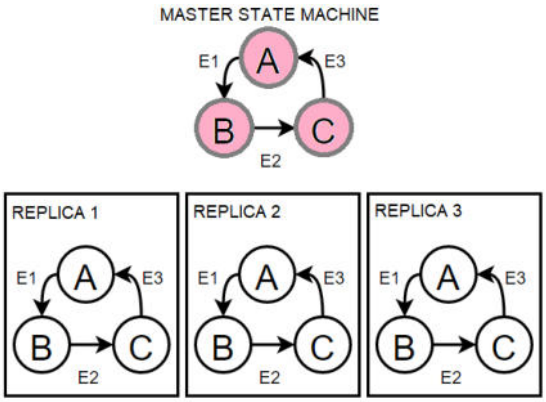


Fig. 1.1: State machine replication

In this study, we utilise characteristics specific to state machines to introduce a distributed checkpointing approach for RSMs (DCfRSM) that simply distributes the checkpointing operation and checkpoint information among replicas. Our approach makes it possible to utilise different levels of replication of checkpointing information to leverage recovery overhead and fault tolerance. We use a simple state machine instance, implemented using spring state machines to demonstrate and evaluate our approach. We use different number of replicas running in a local cluster and in amazon web services separately to measure memory consumption for each replica and recovery time for a booting state machine replica. We compared our approach with two different approaches, namely conventional checkpointing and centralised checkpointing, to evaluate the advantages in using distributed checkpointing for RSMs. Results from these experiments show that DCfRSM provided advantage in terms of memory consumption compared to centralised conventional approaches. On the other hand, DCfRSM produces a high recovery time when it is compared to centralised and conventional approaches because of the extra communication overhead needed for collecting partial histories from different replicas inside a cluster. However, we believe this overhead is the cost of obtaining a higher level of fault tolerance especially with respect to centralised checkpointing. This study expands our earlier preliminary study [4] by providing implementation details of the DCfRSM approach and results from experiments on a real cloud environment.

The rest of the paper is organised as follows: In Sect. 2 we review related literature. In Sect. 3 we explain the DCfRSM approach in more detail. Experimental architecture, implemented approaches for benchmarking purposes and simulation environment are introduced in Sect. 4. Section 5 presents the results of the experiments and the paper is concluded in Sect. 6.

2. Related Work. There exists numerous studies to optimise performance and recovery costs of checkpointing approaches in distributed computing. In this study, we study on application level checkpointing which is applied in a more software-agnostic way by relying on operating on memory as a whole; a comparative discussion between system and application level checkpointing can be found in [5]. The work depicted in [6] states the necessity for replicated state machines to guarantee that majority of replicas inside a cluster can communicate with each other and be prone to node failures. In case of failures on physical machines, in order to minimise checkpointing costs, [7] proposes a novel replication technique with the aim of decreasing recovery costs while [8] proposes a new approach for reducing storage costs.

Due to the review by [9], several layers of fault-tolerance may be defined, such as optimistic fault-tolerance and conservative fault-tolerance mechanisms. This study also states that, by using checkpointing and redo mechanisms, there is a strong chance for ensuring replica consistency for the RSM clusters. Achieving replica consistency, is also one of the features proposed by the DCfRSM approach.

The idea behind using replicated state machines in order to model distributed checkpointing approach is already stated by [2] and [10]. Replicated state machines can be made fault-tolerant with feeding the same

inputs to multiple computers which is the approach used as fundamental principle within scope of experiments of this study. In this aspect, another interesting study is abortable state machine replication approach [11], implemented as an extension for Zyzzyva [12] where authors provide a byzantine fault tolerance mechanism to support interruption of execution in replicated state machines.

Moreover, the study in [1] states that increasing the quality of the user experience is highly dependent on making systems replicated across geographically by using replicated state machines. As suggested by [1], we also experiment on cloud systems to be able to examine the advantage of DCfRSM approach on geographically distributed and replicated state machines.Thesis study in [13], also represents an efficient logging mechanism along with an efficient checkpointing model, which is executed in a parallel and distributed manner by executing concurrent commands. By using this approach, not only recovery process is parallelised but also checkpointing is persisted concurrently in all replicas.

Following geographical distribution, a number of studies has also been publishing the utilisation of checkpointing in cloud environments and in state machines running on cloud environments. Providing checkpointing, as a cloud service has been proposed in an earlier study, where authors have used existing software packages to implement checkpointing on cloud environments [14]. A later study examined checkpointing in edge cloud scenarios and provided algorithms to improve persistence and recovery server selection processes [15]. Having a similar domain with edge domain, a past study uses state machine models of internet of things devices to select optimal points for checkpointing and try to reduce energy overhead of checkpointing process [16].

Another area of literature, regarding checkpointing in the cloud, consists recovery processes of distributed running tasks. A recent study proposes a system in this aspects and evaluates over energy consumption, service level agreement violations, recovery time of tasks and failure rates [17]. A very recent study also reports storage checkpoint recovery times for bag-of-tasks jobs over Amazon Web Services [18]. Even though not being in cloud domain, there has been past research on modelling tasks as state machines and using state machine properties to schedule checkpointing process to optimise restoration time [19].

Reduction of communication between replicas is not the primary concern of our study but there are studies in literature focusing on networking aspects. For instance, the study in [20] proposes a high-performance replicated state machine checkpoint and recovery approach inspired by Paxos consensus protocol. There are also other studies that utilise Paxos such as [21], where authors propose an efficient implementation for snapshots and recovering current state of the state machine.

In order to minimise overall checkpoint overhead, the study in [22] proposes to checkpoint only straggling tasks in order to minimise the number of checkpoints. Within the scope of our study, instead of persisting checkpoints after only certain tasks, as suggested by [22], we have chosen to persist checkpoints to be triggered just after every task execution inside state machines.

With the aim of reducing the checkpoint data size, the study depicted in [23] propose a novel checkpointing mechanism by modelling a decision algorithm in order to reveal the and persist dirty pages that are modified since last checkpoint time. We employ the time ticks and execution history elements in incremental checkpointing approach introduced by [23] and store events occurred within predesignated time ticks for a predesignated execution steps, instead of forcing all the replicas to checkpoint all the modifications performed on the internal states so far. Another study in [24] presents concurrent replication technique for the replicated state machines and compensate non-determinism with the help of static analysis. In our study we reduce the checkpointing storage costs by trying to eliminate redundant checkpointing information from replicas.

Another important aspect of implementing a checkpointing approach is the system level which the designated approach is going to operate on; such as in user level [25] or kernel level [26]. The approach introduced in this paper operates as user level. In the study depicted by [25], states that the user level checkpointing is performed explicitly by external applications and hence, user-level application is unaware whether it is check-pointed or not.On the other hand, the study in [26] proposes an innovative approach called buffered co-scheduling which is implemented at kernel level, hence has unrestricted access to hardware and software resources easily so that operating system's signal mechanism can easily be used for checkpointing formulations.

Although considerable amount of work has been performed on memory checkpointing, very few recent studies exists that provides an approach utilising state machines. A very recent example to such a study uses checkpointing in persisting distributed legacy in memory software by the introduction of a persistent memory

based tool [27]. Another recent study in non volatile memory systems uses differential checkpointing to leverage energy efficiency [28]. Even though state machines are not explicitly used in this study, a recent application of differential checkpointing is presented.

Checkpointing in in-memory processing has been focus of recent studies; an example to such a study is the idea of applying probabilistic checkpointing on the domain of stream processing where authors present a periodic multi-level checkpointing approach and evaluated their approach by experimenting on Apache Flink [29]. An earlier study proposes asynchronous checkpointing approach to be used in in-memory database systems by defining virtual consistency points in application run and apply checkpointing regarding those points [30]. Frequency of checkpointing, lately has drawn some attention as well; a recent study explores how recomputing some data values instead of recovering a persistent copy may decrease checkpointing frequency and provide energy efficiency [31].

Besides checkpointing approaches, there has also been interest in recovery mechanisms with respect to state machine execution context. Due to the study in [32], there are various industry-standard tools which adopts recovery approach in the context of state machines in different aspects. An example to such an approach is "declarative system update", that works by defining the desired state of system and applying necessary modifications to current state in order to achieve the desired state by using RSMs in different context. In addition to this study, the study in [13] increases the performance of the recovery of failed replicas by parallelising the checkpointing operation. Parallelisation, in this study, is achieved by execution of concurrent commands under coordinated and uncoordinated modes of execution. This approach provides a chance for achieving consistency for both faulty and regular(non-faulty) replicas. The study depicted in [33], proposes three novel recovery approaches that produce less overhead during restoration in faulty replicas. Our proposed approach is also inspired from this study in reducing the amount of overhead produced by replicas by reducing the amount of extra processing related to checkpointing.

Efficient checkpointing and recovery mechanisms in the context of replicated state machines has other application areas as well. An example to such an application area can be found in [34], where efficient recovery execution is implemented in the presence of arbitrary faults. The study depicted in [35], proposes to use divide-and-conquer approach for the fault-tolerant replicated state machine cluster systems. According to the study in [36], a decision system inside state machine cluster may predict the executing process being CPU-bound or I/O-bound. According to this decision, subsequent modifications are speculatively executed and used in checkpointing. If the speculation is correct, then checkpoint is made durable and persistent, otherwise, RSM cluster rolls back to previous state to the checkpoint and re-execute further operations for ensuring durability. This approach is stated as beneficial if the time interval of checkpointing is less than the time interval of performing operation which generates the expected result.

3. Distributed Checkpointing for Replicated State Machines (DCfRSM). Distributed checkpointing approach employs deploying and serialising request history handled by an RSM into many pieces during persisting checkpoints. This way, each RSM instance may store a specific piece of history instead of full execution history. During a recovery, whole history is going to be gathered from the components of the system, which also means, logical master history will be shared between all the active state machine replicas.

Definition 3.1 (State Machine). *A state machine is composed of a triplet where S is a set consisting the states in the system, E is the set of events and F is a transition function that represent transitions between the states, each triggered by an event.*

$$M = \{S, E, F\}$$
$$S = \{s_0, s_1, \ldots\}$$
$$E = \{e_0, e_1, \ldots\}$$
$$F \subset S \times E \times S$$

To explain the distributed checkpointing approach in more detail we employ a labelled state transition model where a state machine is defined with a triplet $M = \{S, E, F\}$ such as in Definition 1. In this model S represents the set of states in the transition system, E corresponds to the set of labels used to label the

transitions between the states and F is the transition function between the states that defines a deterministic system.

More precisely, each and every RSM instances includes some states stated as $s_i \in S$, some labels corresponding to events $e_i \in E$ triggering transitions between state machine states such as $s_i \xrightarrow{e_k} s_j$. The transition function F is defined over $S \times E \rightarrow S$. For instance for the master state machine in Fig. 1.1, $S = \{A, B, C\}$ where $E = \{E1, E2, E3\}$ and $F = \{(A, E1, B), (B, E2, C), (C, E3, A)\}$.

We may use the generic machine definition in presented in Definition 1 to demonstrate the distributed checkpointing process. Whole execution history for the RSM instance can be illustrated as in definition in Eq. 3.1. By using this equation, it is possible to state the history begins with a designated state, execution of state machine continues by events that trigger the machine to transit between the states.

$$H = (s_i, e_i, s_j, e_j, s_k, e_k, s_m) \dots \tag{3.1}$$

An execution history instance contains some number of events that results in the machine to transit between states in an orderly manner. In order to represent this order of events we may use superscripts to annotate our history definitions such as in Eq. 3.2[2]. Here, we omit the subscripts that distinguish between the specific events/states for simplicity. In case of a replicated state machine, eventually, each replica of the master state machine is supposed to execute the same order of events. Hence, in a synchronisation agnostic manner, we may distribute the responsibility of saving specific parts of history to specific machines.

$$\begin{aligned} H^{[0-59]} &= (s^0, e^0, s^1, e^1, s^2, e^2, \dots, s^{59}, e^{59}, s^{60}) \\ H_0^{[0-19]} &= (s^0, e^0, s^1, e^1, \dots, s^{19}, e^{19}, s^{20}) \\ H_1^{[20-39]} &= (s^{20}, e^{20}, s^{21}, e^{21}, \dots, e^{39}, s^{40}) \\ H_2^{[40-59]} &= (s^{40}, e^{40}, e^{41}, e^{42}, \dots, e^{59}, s^{60}) \end{aligned} \tag{3.2}$$

A very straightforward example would be the one in Eq. 3.2, where the history is divided into three equally length parts. A division like in Eq. 3.2 might be accomplished in the presence of three replicas which has executed 60 events so far. Each replica saves a specific portion of history which might be represented as H_i^τ where i represents the replica id and τ represents the time interval which replica needs to save the history for the checkpointing purpose. Once the τ is parameter is determined for the overall system, a specific replica might simply perform history saving decision by a simple arithmetic operation. For instance for a three replica system where τ is designated as 20, the replica with id 0 should begin saving history for 20 events every time the modulus of the event number divided by τ equals its own id. Equation 3.3 formalises this calculation by representing replica id by r_{id}[3], event number by e_i and number of replicas by $|R|$ where R corresponds to the replica set.

$$r_{id} == ((e_i \texttt{ div } \tau) \texttt{ mod } |R|) \tag{3.3}$$

In this decision process, an important aspect would be the necessity to broadcast and synchronise whenever a new replica joins the replica set or a present replica leaves the replica set since those situations change the specific points in history where a replica starts saving history for checkpoint. Another important aspect is distributing the history portions in the aforementioned way works correctly for the case when a new replica joins the system but in case of a failure, the specific portion of the history saved by the failing replica becomes lost. In order to deal with this issue, an additional parameter may be introduced to the system such as ρ that represents the replication factor.

Replication factor parameter designates how much each portion of the execution history is replicated among replicas. When performing history saving decision, each replica checks the replication factor parameter as well to starts saving history. For instance, for the straightforward example above ρ is 1 since each portion of the history is saved by a single replica. If we designate ρ as 2 then each portion should be saved by two replicas.

[2]We use simple brackets to represent an ordered set.

[3]We assume replica id's start from 0 in the context of this paper

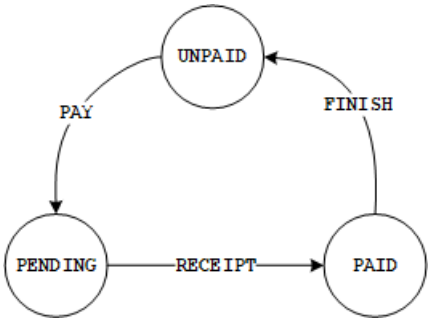


Fig. 4.1: Example book store state machine.

Parameter ρ can be adapted as in Eq. 3.4 simply by adding ρ number of additional condition where the right hand side of the equation is incremented once for each additional condition.

$$\bigwedge_{k=0}^{\rho} (r_{id} == ((e_i \texttt{ div } \tau) \texttt{ mod } |R|) + k) \tag{3.4}$$

For instance, for a 240 event execution, if the number of replicas is 6, τ is determined as 20 and ρ is determined as 2, replica with id 4 is going to save history for twenty events beginning from 80^{th}, 100^{th}, 200^{th} and 220^{th} events. Additionally, replica with id 5 is going to save history for twenty events beginning from 100^{th}, 120^{th}, and 220^{th} events. Since we assume the execution history consists of 240 events, replica 5 is going to stop saving at the end of 240^{th} event but for a case with longer execution histories it is going to continue saving for 20 events starting from 240^{th} event as well. This approach is similar to mirroring and striping approach used in RAID 1+0 implementations [37].

A final remark might be to note that ρ parameter should not exceed the number of replicas, naturally. This parameter provides full history saving by all the replicas when it is set to the number of replicas and provide minimal level of reliability when it is set to 2. If ρ parameter is set to 1, distributed checkpointing will be useful only for the joining replicas to the replica cluster but it will be unreliable in case of a replica failure. It should also be noted that as ρ gets larger it will produce more overhead on each replica during checkpointing and recovery operations.

4. Experimental Environment.

4.1. Overall Architecture. We use a simple book store state machine, as shown on Fig. 4.1 to carry out experiments on distributed checkpointing approach. When a book is ready to be bought from customers, it starts with `UNPAID` state and waits the `PAID` event to be triggered. When the booking and payment operations are performed on the book, `PAID` event is triggered and state has been changed from `UNPAID` to `PENDING` state. In this state, book store waits the receipt from customer in order to ship the book. Once the receipt is received by book store, `RECEIPT` event is triggered and state is transited from `PENDING` to `PAID`. In our experiments we use an implementation of this state machine using Spring State Machines.

We set an experimental environment up using containers and a message queue as illustrated in Fig. 4.2. Each state machine is implemented using Spring State Machine framework inside containers running replicas of the book store state machine. Coordinator node is responsible from generating workload for state machine replicas and coordinating booting sequences of the state machine instances by communicating with replica agents through a simple message queue. We also use coordinators and agents to collect information about the run-time measures that we use to evaluate the performance of the experimented approaches. We have used this architecture in our local experiments as well as cloud experiments.

A typical execution of an experiments kicks off with booting all the replicated state machines inside current cluster. During their booting sequence, replicas prepares themselves to process events -initialize local and shared

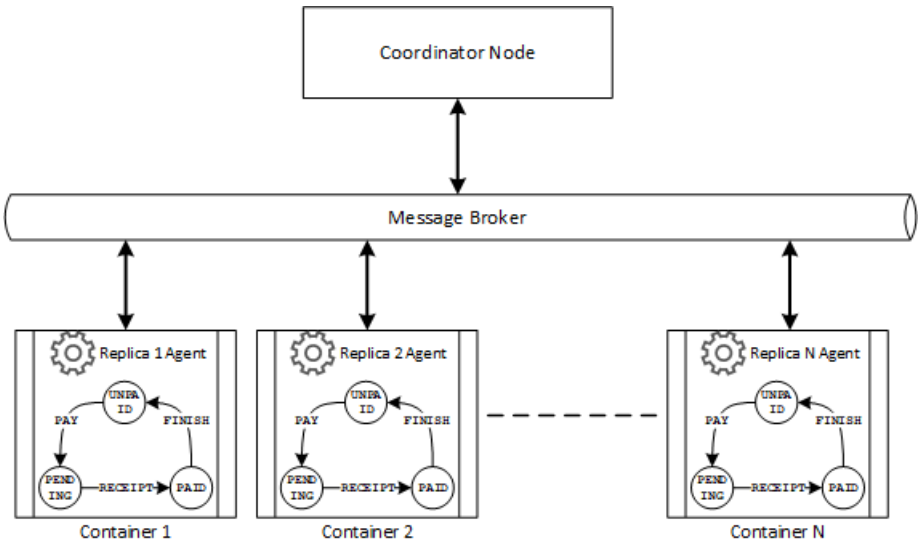


Fig. 4.2: Overall architecture.

variables and so on-, and then, begin listening to incoming events in order to perform transitions between state machine states. Controller node, creates necessary events and sends them to replicas via message broker. The message broker in our architecture is responsible from the following actions:

- Event communication between the coordinator node and replicas,
- Acknowledging controller node of replica life cycle,
- Communicating checkpointing information between replicas and the coordinator node,
- Measuring and reporting number of messages passed through during event processing.

Once the first event is send to state machines, it processes this event, performs necessary operations on its variables and finishes its execution in order to process a new event. After event processing finishes checkpointing operations are performed. During our experiments we use replicas to store checkpointing information as well as coordinator node whenever an external entity is necessary for the checkpointing approach. The details of the checkpointing approaches we have implement is explained in more detail in Sect. 4.2. During checkpointing, we store the context of the state machine which involves inputs and outputs of the current state. Inputs consist of incoming event, event timestamp, source state of the state machine while outputs consist of local and shared variables, destination state of the state machine. As a result of checkpointing process in each replica, whole execution history is recorded as a sequence of state machine contexts in replicas and/or coordinator node depending on the applied approach.

4.2. Implemented Approaches. For our experiments we implemented four different approaches to compare the performance of the distributed checkpointing approach. Initially we implemented centralised checkpointing approach where checkpointing information is stored only in the controller node. Afterwards we implemented conventional approach where each replica stores all the checkpointing information. Finally we implemented two variants of our approach: a striped DCfRSM where each replica stores a single portion of the execution history ($\rho = 1$) and a striped and mirrored DCfRSM where each replica stores two portions of the execution history ($\rho = 2$).

In case of centralised approach, none of RSMs store any of the checkpoints; instead all the checkpoint messages are stored by the controller node, ensuring all the events processed by all the replicas are persisted. To avoid the controller node to be a single point of failure, centralised node can also be replicated. We have left implementation and performance evaluation of such a scenario for a future study.

One of the checkpointing approaches that is used for benchmarking DCfRSM approach is conventional checkpointing. In this approach all the replicas perform checkpointing after each and every event processing, storing exactly the same checkpoints information. In case of any failure on any of the RSM instances, all the checkpoint information should be gathered and applied in order to join back to the RSM cluster. Likewise, a freshly booting replica should communicate with a/some running replica(s) to gather checkpoint information. Memory overhead of this approach is expected to be larger than other approaches, since checkpoint snapshots are redundantly stored by replicas.

As explained in Sect. 3, we implemented two variants of DCfRSM, with replication factor(ρ) set to one and two respectively, to reason about the amount of increase in the overhead as the replication factor parameter gets higher. As discussed earlier, employing a higher level of ρ provides more reliable checkpointing and a higher overhead to replicas.

We provide our implementations for the book store state machine[4] and controller node[5] openly hosted in a cloud repository service to make our experiments reproducible by the scientific community.

5. Experimental Evaluation. For the executions of tests, we use two different environments, a local cluster and a cloud based environment. All the experiments are conducted with 4, 6, 8 and 10 replicas in the experimental environment over 10 repetitions for each experiment by sending a total number of 3600 requests for the master state machine of the replicated cluster. We set the replica's history portion interval τ to 120 events being a common multiple for each different number of replicas used in the experiments and also being a divisor of total number of requests used in the experiments.

During these experiments average amount of memory consumption used by all replicas in the cluster is measured as well as average of restore duration of newly joining replica. In order to measure memory consumption of each replica during state machine execution, an external library is used for counting number of checkpoint objects in memory. Java's instrumentation API[6] is used during state machine execution to measure and log memory usage whenever a checkpoint is about to be persisted. An overview of the application of our experiments can be summarised as follows:

- Initialise controller node and message broker,
- Initialise the necessary number of replica in the cluster,
- Trigger messages from controller node, wait for replicas to finish execution,
- Once all the events are processed, compare local and shared variables of all the replicas in order to ensure that replicas executed consistently,
- Boot a new replica in order to join the cluster, wait for the replica to gather checkpoint information from respective node/nodes

Once the new replica finishes its execution, it means that first round of the experiments are finished. As of all the experiments for the respective replica set is finished, reports can be generated. By using the flow above, total memory consumption of the cluster is calculated for the replicas. Then, averages and standard errors for repeated experiments are calculated.

5.1. Experiments on local cluster. As a local cluster we use computers with 2.60 GHz Intel i5 processors, 4 GB RAM and 100 GB SSDs running debian linux distributions. We begin presenting the experiments on our local cluster by examining memory consumption of each approach on average for each replica. Figure 5.1 presents and compares the memory consumption for approaches. As expected, conventional approach constantly consumes the highest amount of memory since all the replicas in the cluster keep the whole history all the time for this approach. Likewise, centralised approach constantly consumes the lowest amount of memory since replicas do not keep any checkpointing history for this approach. Distributed checkpointing approaches stand in the middle between conventional and centralised approaches and spend less memory as the number of replicas increase since the history will be divided among more number of replicas. Comparably, mirroring on top of striping increases the amount of memory consumption, as expected.

For restore duration in Fig. 5.2, the results in the local cluster are close and have high deviations. However,

[4] `https://github.com/celikelozdinc/DistributedStateMachine`
[5] `https://github.com/celikelozdinc/LoadBalancer`
[6] `https://docs.oracle.com/javase/7/docs/api/java/lang/instrument/Instrumentation.html`

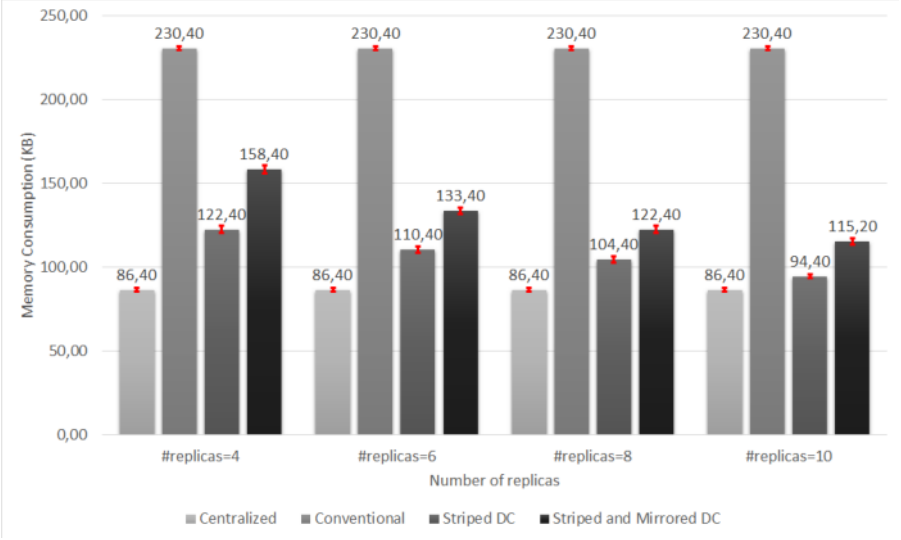


Fig. 5.1: Average memory consumption by replicas.

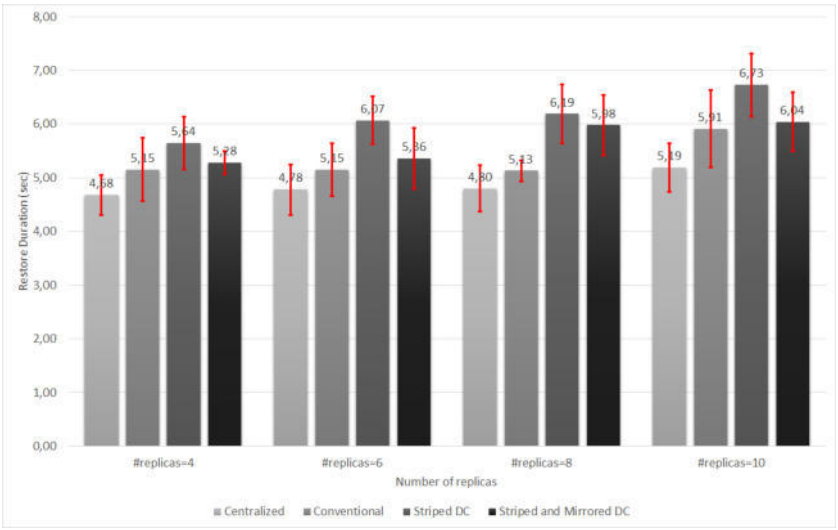


Fig. 5.2: Restore duration.

the average restore duration for centralised approach performed the best with respect to other approaches since the booting sequence requires less communication and only with coordinator node. Conventional approach came the second because obtaining history information from another replica requires and extra step of communication in our implementation compared to centralised approach: during booting sequence checkpoint information needs to be communicated from a running replica to coordinator and then from the coordinator to booting replica. This overhead may be avoided by enabling the booting replica to directly obtain checkpoint data from a running replica. Distributed checkpointing approaches perform worse because they need more communication

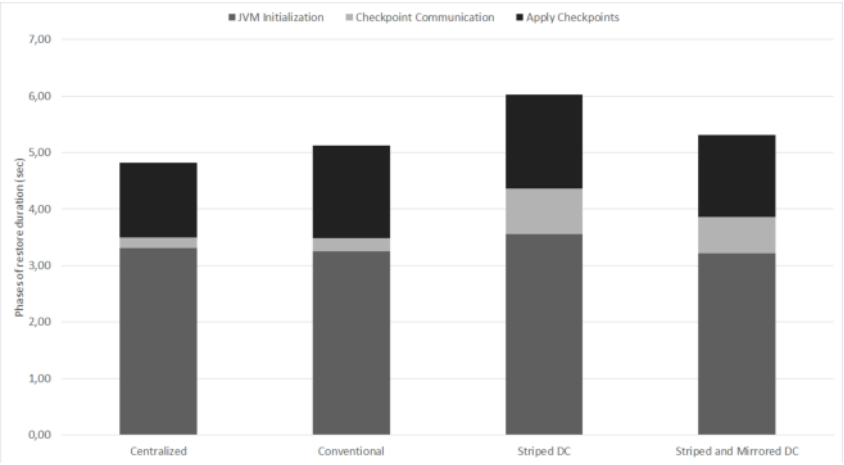


Fig. 5.3: Average time spend for each phases of restoring

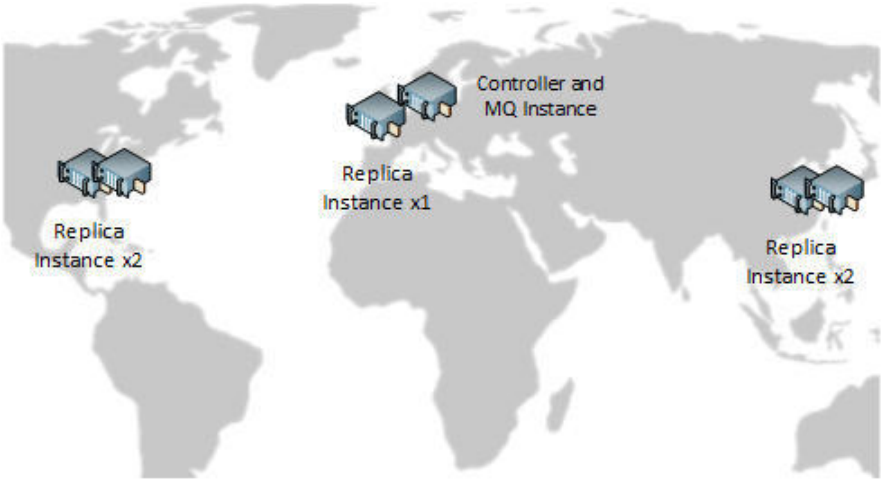


Fig. 5.4: AWS EC2 architecture for cloud experiments.

with other replicas more than the conventional and centralised approaches. An important remark might be the mirrored and striped approach beating the striped approach for restore duration. We believe, this is due to the more number of alternatives to obtain checkpoint data portions during restoring phase. Any slow responding, bottleneck, replica is eliminated due to the presence of alternatives to obtain the same data when striping and mirroring is applied.

Furthermore, we investigated the time spent during the restoring of a booting replica in Fig. 5.3. It can be seen that the difference between different approaches is greatly due to the checkpoint data communication phase.

5.2. Experiments on cloud environment. We also repeat our experiments on geographically distributed `t3.medium` instances running on a Amazon Web Services Elastic Compute Cloud (AWS-EC2). As per Fig. 5.4, 6 virtual machines from 3 different regions are used for executing experiments in cloud environment. While executing experiments, controller node and message broker service is isolated and positioned on a dif-

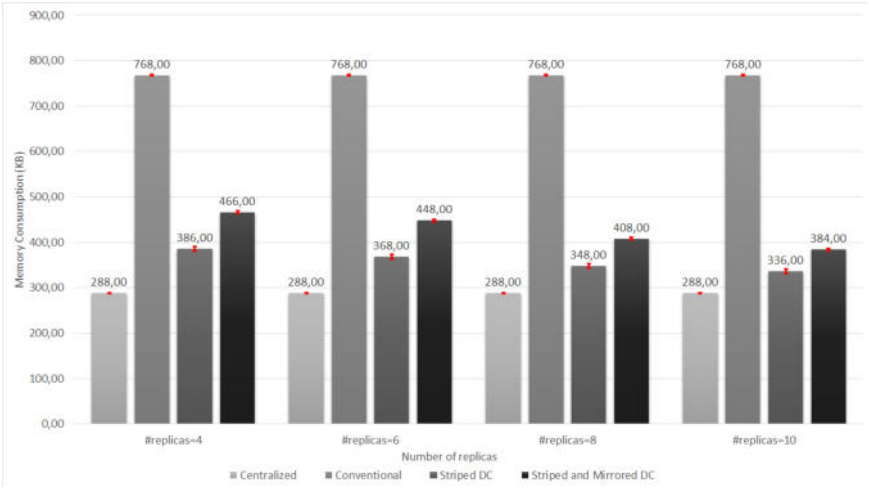


Fig. 5.5: Memory consumption from the experiments in cloud.

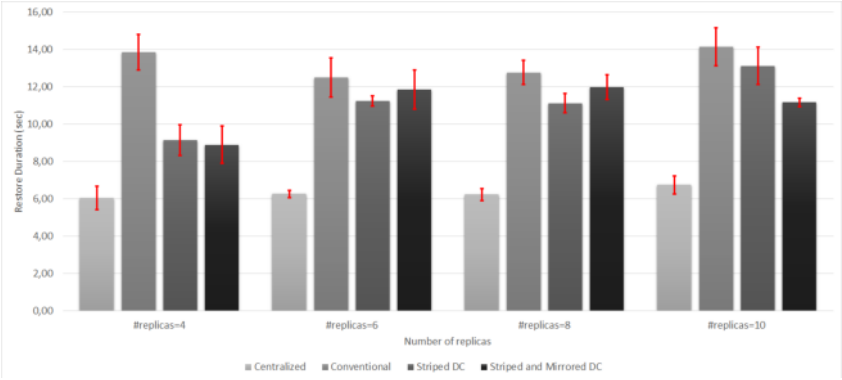


Fig. 5.6: Restore duration from the experiments in cloud.

ferent region which is totally apart from RSM instances. All the RSM instances distributed among 4 virtual machines and hence, spread to 2 regions. Freshly booting replicas are joint to cluster from a region apart from the regions of active replicas. By doing so, whenever a new replica joins the cluster, it is needed to gather checkpoint snapshots from different machines on geographically distributed regions.

Figure 5.5 shows the same advantage of DCfRSM approach in terms of memory consumption. Distributed approaches spend less memory since they divide the history data to multiple parts during their execution. Centralised approach is the best in this respect, naturally, since it doesn't require any replica to keep any checkpointing data.

As per Fig. 5.6, results from experiments in cloud environment shows some differences in terms of restore duration in cloud environments. Conventional approaches perform worse than the rest due to the fact that the booting replica is always in a geographically different region than the replica that provides checkpointing information. Likewise, for the centralised approach it is guaranteed that the booting replica and the coordinator are in the same region, which provides an advantage of communication latency during the booting time. In more realistic scenarios these measurements might change form case to case. A booting replica might not always find the checkpoint data in a close replica in terms of geographical location or network latency. However,

our experiments provide the best and worst case scenarios under centralised and conventional approaches respectively to show the place of distributed checkpointing approaches with this respect. For distributed checkpointing, restore duration is always better than conventional approach even though the booting replica is in a different region than all the other replicas. This situation is due to the exploitation of alternatives instead of relying on a single replica. On the other hand, for cloud experiments, striped approach has performed slightly better than the striped and mirrored approach for small number of replicas but striped and mirrored approach performed much better as the number of replicas reached to 10. This situation shows that for increased network latency mirroring might lose its positive effect on restore duration for small number of replicas.

6. Conclusion and Future Work. In this paper, distributed checkpointing for the replicated state machines is examined and compared with conventional approaches. Especially in terms of full replication of checkpointing data and using a single node, distributed checkpointing approaches provide a mediation point to leverage between the amount of fault tolerance of the cluster versus restore duration of the replicas. Our experiments show that using distributed checkpointing provides a certain amount of memory consumption advantage and provides worse (as expected) but comparable restore duration. Main advantage of using a distributed checkpointing approach is to distribute the checkpointing information among replicas to provide an adjustable level of fault tolerance during replicated state machine execution.

Our studies can be extended to decrease recovery time overhead as much as possible in order to provide a better trade-off between distributed checkpointing and other approaches. Though many possible improvement opportunities exist in our implementations, allowing replicas to communicate each other via agents to eliminate the need to use a coordinator node might the most important one. Another possibility might be to better parallelise the recovery phase for distributed checkpointing since the approach benefits from using independent portions of the execution history. Moreover, various different values for parameters τ and ρ might be used to find optimal values for different scenarios in RSM checkpointing. Finally, providing the reliability of history portions by using parity information instead of mirroring might be another possible improvement to store even less checkpointing information in this context.

REFERENCES

[1] C. Li, D. Porto, A. Clement, J. Gehrke, N. Preguiça, R. Rodrigues, Making geo-replicated systems fast as possible, consistent when necessary. *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)*, pages 265-278, 2012.

[2] W.J. Bolosky, D. Bradshaw, R.B. Haagens, N.P. Kusters, P. Li, Paxos replicated state machines as the basis of a high-performance data store. *Proc. NSDI'11, USENIX Conference on Networked Systems Design and Implementation*, pages 141-154, 2011.

[3] J. Du, D. Sciascia, S. Elnikety, W. Zwaenepoel, F. Pedone, Clock-RSM: Low-latency inter-datacenter state machine replication using loosely synchronized physical clocks. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 343-354, IEEE, 2014.

[4] N.Ö. Çelikel, T. Ovatman, A Distributed Checkpoint Mechanism for Replicated State Machines. *Proceedings of the 10th International Conference on Cloud Computing and Services Science, CLOSER 2020*, Prague, Czech Republic, May 7-9, 2020, pages 515-520, SCITEPRESS, 2020.

[5] Posner, J. System-Level vs. Application-Level Checkpointing. *IEEE International Conference on Cluster Computing (CLUSTER)*, pages 404–405, 2020.

[6] R. Friedman, A. Vaysburd, Fast replicated state machines over partitionable networks. *Proceedings of SRDS'97: 16th IEEE Symposium on Reliable Distributed Systems*, pages 130-137, 1997.

[7] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, A. Warfield, Remus: High availability via asynchronous virtual machine replication. *Proceedings of the 5th USENIX symposium on networked systems design and implementation*, pages 161-174, San Francisco, 2008.

[8] J. Heo, S. Yi, Y. Cho, J. Hong, S.Y. Shin, Space-efficient page-level incremental checkpointing. *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1558-1562, 2005.

[9] W. Zhao, Performance optimization for state machine replication based on application semantics: a review. *Journal of Systems and Software*, 112:96-109,2016.

[10] F.B. Schneider, Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*,

[11] Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M., The next 700 BFT protocols. *ACM Transactions on Computer Systems*, Vol. 32, No. 4, pages 12:1–12:45, 2015.

[12] Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E., Zyzzyva: speculative byzantine fault tolerance. *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages=45–58, 2007.

[13] O.M. Mendizabal, Fast recovery in parallel state machine replication. *Pontifícia Universidade Católica do Rio Grande do Sul*, 2016. 22(4):299-319, 1990.

[14] Cao, J., Simonin, M., Cooperman, G., Morin, C., Checkpointing as a service in heterogeneous cloud environments. *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 61–70, 2015.

[15] Zhou, A., Sun, Q., Li, J., Enhancing reliability via checkpointing in cloud computing systems. *China Communications, IEEE*, volume 14, number 7, pages 1–10, 2017.

[16] Mirhoseini, A., Rouhani, B. D., Songhori, E., Koushanfar, F. Chime: Checkpointing long computations on interm ittently energized iot devices. *IEEE Transactions on Multi-Scale Computing Systems*, volume 2, number 4, pages 277–290, 2016.

[17] Meroufel, B., Belalem, G., Optimization of checkpointing/recovery strategy in cloud computing with adaptive storage management. *Concurrency and Computation: Practice and Experience*, volume 30, number 24, pages e4906, Wiley Online Library, year=2018.

[18] Teylo, L., Brum, R. C., Arantes, L., Sens, P., Drummond, L. M. D. A., Developing Checkpointing and Recovery Procedures with the Storage Services of Amazon Web Services. *49th International Conference on Parallel Processing-ICPP: Workshops*, pages 1–8, 2020.

[19] Levitin, G., Xing, L., Luo, L., Joint optimal checkpointing and rejuvenation policy for real-time computing tasks. *Reliability Engineering & System Safety, Elsevier*, volume 182, pages 63–72, 2019.

[20] M. Yanhua, P.J. Flavio, M.Keith, Mencius: building efficient replicated state machines for WANs. *8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 08)*, 2008.

[21] J. Konczak, N.F. de Sousa Santos, T. Zurkowski, P. Wojciechowski, A. Schiper, JPaxos: State machine replication based on the Paxos protocol. *EPFL- I&C - School of Computer and Communication Sciences, LSR - Distributed Systems Laboratory, Technical Report*, No. REP_WORK, 2011.

[22] B. Ghit, D. Epema, Better safe than sorry: Grappling with failures of in-memory data analytics frameworks. *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, pages 105-116, 2017.

[23] N. Naksinehaboon, Y. Liu, C. Leangsuksun, R. Nassar, M. Paun, S.L. Scott, Reliability-aware approach: An incremental checkpoint/restart model in hpc environments. *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 783-788. IEEE, 2008.

[24] J.G. Slember, P. Narasimhan, Static Analysis Meets Distributed Fault-Tolerance: Enabling State-Machine Replication with Nondeterminism. *HotDep*, 2006.

[25] J.C. Sancho, F. Petrini, G. Johnson, E. Frachtenberg, On the feasibility of incremental checkpointing for scientific computing. *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings*, page 58, IEEE, 2004.

[26] R. Gioiosa, J.C. Sancho, S. Jiang, F. Petrini, Transparent, incremental checkpointing at kernel level: a foundation for fault tolerance for parallel computers. *SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, pages 9-9, IEEE, 2005.

[27] Zhang, W., Shenker, S., Zhang, I., Persistent State Machines for Recoverable In-memory Storage Systems with NVRam. *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pages 1029–1046, 2020.

[28] Ahmed, S., Bhatti, N. A., Alizai, M. H., Siddiqui, J. H., Mottola, L., Efficient intermittent computing with differential checkpointing. *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, pages=70–81, 2019.

[29] Jayasekara, S., Harwood, A., Karunasekera, S. Optimal Multi-Level Interval-based Checkpointing for Exascale Stream Processing Systems. *arXiv preprint arXiv:1912.07162*, 2019.

[30] Ren, K., Diamond, T., Abadi, D. J., Thomson, A., Low-overhead asynchronous checkpointing in main-memory database systems. *Proceedings of the 2016 International Conference on Management of Data* pages 1539–1551, 2016.

[31] Akturk, I., Karpuzcu, U. R. ACR: Amnesic Checkpointing and Recovery. *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 30–43, 2020.

[32] T. Kuwahara, T. Kuroda, M. Nakanoya, Y. Yakuwa, Y. Sato, Y. Matsunaga, Automated Planning of System Rollback in Declarative IT System Update. *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 428-434, 2019.

[33] J.Z. Konczak, P.T. Wojciechowski, N. Santos, T. Zurkowski, A. Schiper, Recovery Algorithms for Paxos-based State Machine Replication. *IEEE Transactions on Dependable and Secure Computing*, 2019.

[34] J. Rushby, Reconfiguration and transient recovery in state machine architectures. *Proceedings of Annual Symposium on Fault Tolerant Computing*, pages 6-15, IEEE, 1996.

[35] F.B. Schneider, L. Zhou, Implementing trustworthy services using replicated state machines. *IEEE Security & Privacy*, 3(5):34-43, 2005.

[36] B. Wester, J.A. Cowling, E.B. Nightingale, P.M. Chen, J. Flinn, B. Liskov, Tolerating Latency in Replicated State Machines Through Client Speculation. *NSDI*, pages 245-260, 2009.

[37] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson, RAID: High-performance, reliable secondary storage. *ACM Computing Surveys (CSUR)*, 26(2):145-185, 1994.

Edited by: Dana Petcu
Received: Dec 3, 2020
Accepted: Jan 21, 2021

Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 1, pp. 81–92, DOI 10.12694:/scpe.v22i1.1847

IMPROVED LOCALIZED SLEEP SCHEDULING TECHNIQUES TO PROLONG WSN LIFETIME

NACHIKETA TARASIA* AMULYA RATNA SWAIN† SOHAM ROY‡ AND UDIT NARAYANA KAR§

Abstract. A standard Wireless Sensor Networks(WSNs) comprises of low-cost sensor nodes embedded with small batteries. To enhance the network lifetime of WSN, the number of active nodes among the deployed nodes should be minimum. Along with this, it must be ensured that coverage of the targeted area would not get affected by the currently active nodes. Considering different applications of WSN, there is still a demand for full coverage or partial coverage of the deployed area. Irrespective of the circumstances, a proper sleep scheduling algorithm needs to be followed. Else, the active nodes will be tuckered out of the battery. Random distribution of the sensor nodes in a common area may have multiple active nodes. It is essential to identify the redundant number of active nodes and put them into sleep to conserve energy. This paper has proposed a methodology where the active sensor nodes form a hierarchical structure that heals itself by following a level-wise approach. In the meantime, it also detects the total number of redundant nodes in the coverage area. The performance of the proposed protocol is evaluated using the Castalia simulator. The simulation results show that the proposed level-wise periodic tree construction approach increases the network's durability in conjunction with the level wise approach.

Key words: Wireless Sensor Network, Sleep Scheduling, Coverage, Energy Efficient.

AMS subject classifications. 68M14

1. Introduction. Wireless Sensor Networks (WSNs) have been generally considered as one of the most imperative innovations. A regular Wireless Sensor Network comprises many low-cost devices known as sensor nodes. These multi-practical sensor nodes have limited battery life and are generally used to monitor a region of interest [1] [2]. These sensor nodes have inbuilt miniaturized controllers and radio transceivers. Subsequently, sensor nodes can detect outside events, process the detected information, and transmit it to the sink. WSNs are broadly utilized for ecological condition monitoring, security surveillance of combat zones, monitoring untamed natural life, etc. [3]. Sensor nodes, as a general, are densely deployed in an inhuman domain, where it would be tough to maintain the nodes' battery capacity. To observe and control the physical conditions, WSNs should address the following two requirements: (i) sensing in the intended zone should be appropriately done, and (ii) proper communication should be maintained among the sensor nodes to ensure that the collected data is properly transmitted to the sink node. Else, the overall collection of the data is pointless.

In WSN, a sensor node can sense its detecting range, which is called as sensing coverage of the node. The network coverage [3] [4] could be translated as the aggregate coverage by all the *ACTIVE* sensor nodes. Similarly, an *ACTIVE* node should send the information to another node within its radio coverage area. More is the number of *ACTIVE* nodes; more is the consumption of energy. To boost the lifetime of the WSN, it is logical to limit the number of *ACTIVE* nodes while accomplishing the most extreme conceivable sensing and radio coverage.

Depending upon the necessities of the application, the sensing and radio coverage may be limited [5]. For an application like intrusion detection or movement detection, it must have at least one *ACTIVE* sensor node for each location. So, most of the *ACTIVE* nodes will run out of battery very rapidly. Similarly, for applications like humidity or temperature monitoring, it is required to have fewer numbers of *ACTIVE* nodes, with limited coverage. Nevertheless, if any node is *ACTIVE* for a long duration, it is necessary to have a proper sleep

*KIIT Deemed to be University, Bhubaneswar, India (ntarasia06@gmail.com).
†KIIT Deemed to be University, Bhubaneswar, India (swainamulya@gmail.com).
‡KIIT Deemed to be University, Bhubaneswar, India (skr2538@gmail.com).
§KIIT Deemed to be University, Bhubaneswar, India (uditnarayankar@gmail.com).

scheduling mechanism to manage the whole network. Applying a sleep scheduling mechanism over WSN allows the sensor nodes to share their duties among themselves. This mechanism can be constructive in conserving the energy of the sensor nodes. There may be instances where multiple sensor nodes are deployed to cover a common area, which ultimately resulting in wastage of battery. In such scenarios, it is essential to identify the redundant nodes. These redundant nodes can move to sleep mode to enhance the network lifetime. To achieve this, WSNs must follow an appropriate duty cycle mechanism that govern the cycle of slreep and wake-up mode among the sensor nodes.

In this manuscript, we have proposed a sleep scheduling mechanism where redundant nodes are identified and put those nodes into sleep mode to conserve energy. Our proposed protocol ensures full coverage of the desired region. The proposed approach uses a level-wise hierarchical structure, where the sensor nodes mend themselves locally intending to save energy. The rest of the paper is organized as follows. Section II presents state of the art, followed by a proposed approach presented in Section III. The detailed simulation generated results, and analysis of the generated graphs are presented in Section IV. Finally, the manuscript concludes in Section V.

2. State of the Art. Energy conservation is one of the significant taxonomies of WSNs. To enhance the overall lifetime of the WSNs, energy conservation schemes are being consistently investigated and scrutinized by researchers across the globe. Hence, there is an abundant number of research articles available in the background. There exists a considerable different coverage optimization approaches like probing environment and adaptive sleeping (PEAS), probing environment and collaborating adaptive sleeping (PECAS), controlled layer deployment (CLD), random back-off sleep protocol (RBSP), and so on. In PEAS [6], the network lifetime has been extended by embracing a basic 3-mode approach, i.e., sleeping, probing, and active. For each probing region, an active node has remained in charge of coverage, and in the meantime, the other sensor nodes stay in sleep mode to spare energy. The sleep node goes into the probing stage from time to time and checks for the accessibility of the active node's presence by sending a probing message. At that point, it returns to sleep mode to spare energy. The PEAS has some impediment, i.e., a sensor node, which ended up active, must be in a similar state all the time till it dries out which may result in lopsided vitality utilization in the system. PECAS [7] is the extended version of PEAS, where it overcomes few limitations of the prior. The active node in PECAS goes to sleep mode after a stipulated period, but it shares its remaining energy with its neighbors before it goes to sleep mode. This information is used by the probing nodes in the region to decide when to be active again.

The cascading effect is the most commonly faced issues in WSN between the sink and the leaf nodes. The nodes closer to the sink are engaged in transmitting data most of the time compared to far away from leaf nodes. If any event occurs far away from the sink, then more intermediate nodes participate, thus shortening the network's lifetime. The PEAS algorithm has been modified in CLD [8] approach, which uses deterministic node deployment to counter the cascading effect. In CLD, the average distance between two active nodes is maintained as $2r/3$, where r is the sensing radius. Sleeping nodes surround the active nodes at a distance of $r/6$. More number of sleep nodes are placed near the sink node to overcome the cascading effect. CLD can be implemented in those applications where the target area is known beforehand.

RBSP [9] is a probe-based algorithm that uses information regarding the rest of the energy level of the present active node. Here, back-off sleep time is calculated, which is utilized by the currently active node's neighboring nodes to choose when to wake up and examine the currently active node's status. Using this approach, when an active node has high outstanding energy, a neighboring node's chance to turn active is low and the other way around.

The authors [28] recommended sleep scheduling algorithm is intended to improve network administration by reducing power distribution due to the passive listening of nodes. The sleep period is proportionate to the remaining power of sensor nodes and adaptive. A sleep scheduling approach is required to adjust the network administration by utilizing the least energy. A distributed sleep scheduling system allows the sensor node to entirely satisfy the sensing ranges and switch off the node if the conversation doesn't occur or does not have adequate energy [29].

One of the significant challenges in formulating such systems lies within the obliged energy and computational assets accessible to sensor nodes. These constraints must be taken into consideration at all levels of the

system progression. The arrangement of sensor nodes is the primary step in setting up a sensor network. Since WSNs contain many sensor nodes, the nodes must be placed in clusters [30], where the area of each specific node cannot be wholly ensured a priori. In this manner, the number of nodes deployed to cover the complete observed zone is mostly higher. The authors in [10] have presented an algorithm that chooses mutually elite sets of sensor nodes, where the union of these sets covers the observed region. The interim actions are same for all collections, and one of the groups is active. This algorithm accomplishes considerable energy savings along with ultimately protecting coverage.

In [11], the authors defined the coverage issue as a choice based problem. Here, the objective is to decide whether each point within the desired region of the sensor network is secured by at least k nodes, where k could be predefined esteem. The sensing ranges of nodes can be unit disks or non-unit disks. This paper displayed polynomial-time calculations in terms of the number of nodes, which can be effectively interpreted in distributed conventions. The simulation result showed that energy could be conserved along with the fault-tolerant model in an area where nodes are deployed randomly. In [12], the authors address the difficulty of choosing the least number of associated sensor nodes to cover a distributed set of interest objects. A centralized algorithm runs by iteratively appending nodes that maximize a measure called k-benefit to an initially empty set of nodes. Though running with the least number of sensor nodes does not singularly signify the system's maximal lifetime. Without global optimization, any nodes that can cover many targets could be recorded to work massively, and they will soon run away from energy.

In [13], the authors disseminated a single step arrangement algorithm that divides the region of intrigued into two equal networks. The nodes are deployed to possess each point in grids to be completely covered and connected. Two strategies have been proposed for the deployment of sensor nodes, i.e., randomized and planned as per the situations. The authors in [14] have proposed an approach in which a moving robot fixes the coverage hole by picking nodes from the coverage area where redundant nodes are present. A carrier-based sensor relocation by robots to mend coverage holes has also been proposed in [15] that uses a virtual force approach in a grid structure of interest. In this approach, full coverage is achieved by placing redundant sensor nodes in coverage holes with the help of robots that randomly move in the network and are restricted in grids.

In [16] and [17], an energy-efficient coverage approach has been proposed where authors consider a large number of sensor nodes were deployed in the area of interest to achieve coverage. In this approach, the number of sensor nodes is highly populated; these nodes are divided into several disjoint sets. The idea of prolonging the network lifetime is by putting the rest of the nodes into sleep mode, whereas one disjoint set is active in a specific area for coverage. This mentioned algorithm follows a centralized approach. A localized and distributed algorithm [18] called Node Scheduling scheme Based on the Eligibility rule(NSBE) that follows a scheduling approach where each node decides to be in either active or in sleep mode. At any instance, one node is active for covering the area of interest, and the redundant nodes are in sleep mode. In each cycle, the active node tries to find an alternative node, which will cover the area of interest. If the alternative node is found, then the active node can go to sleep mode. There is no simulation proof of the stated approach.

Sensor Scheduling for k-Coverage(SSC) [19] monitors a two-dimensional region, where the same locations do not have sensor nodes more than one. K number of sensor nodes must guard each point continuously. The SSC is a centralized algorithm, which is NP-hard. The existence of WSN is determined as the whole span during which the entire region is k-covered. The authors in [20] recommended a solution for domain coverage in a synchronous WSN, where radio strength is equal to the sensing range. This approach considered that each sensor node knows the exact location of its neighbor nodes. The precise location information helps the node to decide autonomously either to be in active or in sleep mode. A node can go to sleep mode if it's sensing range is covered by its neighbor nodes. A backoff algorithm is being followed to avoid coverage gaps while going to sleep mode. However, this may affect the node connectivity.

A centralized algorithm has been proposed by [21], where the sensor nodes are grouped into different sets. The nodes of a particular set are active to cover the required positions. A scheduling algorithm decides the nodes' state of being alternate between active and sleep to extend the whole network's lifespan. An adaptable energy-efficient sensing coverage protocol using a differentiated monitoring service for WSN has been proposed by [22]. Here, each node ensures a certain coverage degree by obeying dynamic scheduling for self. The proposed protocol uses the grid-based approach, where the deployed area is split into several grids, and each

grid is allocated with a sensor node. Each active node covering any grid also maintains a list of other nodes that can also cover the same grid. If the network is dense, then the time and space complexity are high as the list contains the sensor node details.

The author in [23] proposed a greedy algorithm that maximizes the network lifespan by following a state scheduling approach. The state scheduling approach gives autonomy to nodes to become active or inactive initially. The proposed approach then tries to cover the target area with the selected active nodes; if failed, the algorithm restarts. However, the algorithm consumes more energy concerning the exchange of messages [24, 25]. It should be noted that all the readings discussed earlier address the scheduling problem for coverage. But not the coverage problem, which ensures connectivity that we concentrate on to achieve. We analyze the sensor nodes' scheduling obstacle to observe the full coverage and connectivity.

3. Proposed Work. After the initial tree construction with currently selected active nodes, the nodes are categorized into two types, viz. the internal nodes($type_1$) and leaf nodes($type_2$). Together they maintain full coverage of the whole deployment area. So, they are also referred to as the coverage nodes. The internal nodes are always active as they receive data packets from their children and send it to the sink node. Nevertheless, the leaf nodes along with $type_1$ nodes do the job of sensing the event and sending messages to their parent. The leaf nodes stay in sleep mode most of the time. These leaf nodes turn active only when they perform the sense operation in case of an event, pass the message to their parent, and again go back to sleep mode.

From the above description, it can be seen that all the sensor nodes participate both in the event detection process as well as message passing. However, in a WSN with high concentration of nodes, either the sensing radius of sensor nodes are overlapping with each other or a combination of nodes entirely coincides with the sensing area of some other nodes. In such scenario, when all these nodes sense their surroundings and send the sensed data to their parent nodes, they perceive the same occurrence of the event and transmit the same data to the sink, which is entirely unnecessary. This leads to ineffective use of energy, where all the efforts by these nodes are useless at the expense of the tree's longevity.

To ensure full coverage of the deployed area, all the sensor nodes do not have to participate in event detection. Some of the nodes can completely remain in sleep mode and do not take part in any operation, and still the full coverage can be ensured. Such nodes remain a part of the tree but do not contribute in providing coverage. Here, those nodes are referred as the $type_0$ nodes. As long as a node is completely in sleep mode, it is equivalent to a dead node.

In WSNs, the sensing radius of most of nodes overlaps with each other. A node is said to be redundant, if its sensing area is within the sensing range of one or more sensor nodes. For area coverage in WSNs, the basic idea is to find the redundant nodes in a required area, which is already under the coverage of some other nodes, and put those redundant nodes into sleep mode. Keeping longevity and coverage in mind, in this paper, we have proposed a novel approach that initially constructs a hierarchical structure and periodically mends itself locally with consultation of nearby nodes based on nodes' level. It finds the redundant nodes in the vicinity of currently changing area only with the involvement of a minimum number of sensor nodes. Thus, full coverage is ensured only by engaging the nodes of the concerned area without involving all the nodes of the whole tree.

3.1. Construction. Based on initial tree construction, the sensor nodes are categorized into three types viz., $type_0$, $type_1$, and $type_2$ nodes. $Type_0$ nodes are the redundant sleep nodes which are currently in sleep mode and do not partake either in event detection or in area coverage. Their behavior is equivalent to a dead node. $Type_1$ nodes are the internal nodes and always remain active for a fixed duration. $Type_2$ nodes are called non-redundant leaf nodes that remains inactive but take part in the coverage of deployed area. Among these three types of nodes, $type_1$ nodes consumes more energy as compared to $type_0$ and $type_2$ nodes. During periodic tree reconstruction, the $type_1$ nodes, which meet the criteria to go to sleep, will be put into sleep mode, i.e., $type_1$ nodes will be turned to either $type_0$ or $type_2$ nodes depending upon the need of coverage of the locale. Moreover, before the reconstruction phase for the $(n+1)^{th}$ round, it is ensured that the tree was under full coverage and connectivity is maintained till the end of the n^{th} round.

We use the different abbreviations to describe the proposed protocol as given in Table 3.1.

For this proposed protocol, the sink node always remains active during the entire lifetime of the network. Before the beginning of $(n+1)^{th}$ round of tree reconstruction phase, a node $N(i)$, which was considered as

Table 3.1: Abbreviation

Short Name	Description
N(i)	i^{th} node
SS(N(i))	Sleep signal for i^{th} node
RE(N(i))	Remaining energy for i^{th} node
LEVEL(N(i))	Level of i^{th} node
PT(N(i))	No. of packets transfer by i^{th} node in a round
ALIVE(N(i))	True if i^{th} node is alive
CONS(N(i))	No. of consecutive rounds node i is active
CRL(n)	Current level reconstruct for the n^{th} round. CRL(n) = (CRL(n-1) +1) % max_level. A CRL value of 3 means level 1,2,3 will be reconstructed.
CHECK_REDUNDANCY(N(k))	k^{th} node checks for redundancy among its neighbors.

$type_1$ node in the n^{th} tree reconstruction phase and spends a significant amount of energy, can initiate its sleeping process based on satisfying any one of the following criteria.

1. Number of packets transmitted by node $N(i)$ in the current n^{th} round, i.e. represented as $PT(N(i))$, is greater than the threshold value.
2. Number of consecutive rounds the node $N(i)$ is active, i.e. represented as $Cons(N(i))$, is equal to three.
3. The level number of node $N(i)$ is less than or equal to the current level reconstruct for the n^{th} round, which is represented as $CRL(n)$. This $CRL(n)$ value plays an important role as the proposed protocol claims that rather than applying the sleeping process over all the nodes of WSN in each reconstruction phase, it applies on nodes present in certain level of the tree in different reconstruction phase. This is necessary because all the nodes in the different level of the tree do not consume energy in a uniform manner. The nodes which are closer to the sink consume more energy as compared to the nodes which are far away from sink. So, the nodes present in the higher level of the tree need to participate more frequently in tree reconstruction as compared to nodes in lower level. Considering the above necessity, we compute the $CRL(n) = (CRL(n-1)+1)\%max\ level$ that indicates up to which level the nodes will participate in n^{th} round of reconstruction phase. For example, if the CRL value is 3 then it means that nodes in level 1, 2, and 3 will participate in reconstruction phase.

Once, the $type_1$ node N(i) initiates its sleeping process, it sends a sleeping signal called SS(N(i)) to find new parents for all of its children before it is allowed to turn into $type_0$ or $type_2$ node. Then the node $N(i)$ starts with broadcasting a $FIND\_PARENT$ message packet to its children to find their new parent.

As per the algorithm-1, the active node N(i) waits for a random amount of time until all its children get a new parent before it turns into $type_0$ or $type_2$ node depending upon whether it is considered to be a redundant node with the support of its nearby $type_1$ and $type_2$ nodes. Once a node $N(j)$, which is a child of node $N(i)$, receives the $FIND\_PARENT$ packet, it initiates its process to find a new parent by broadcasting a WANT_PARENT packet and waits for a random amount of time to decide its new parent. A node N(k), which receives a WANT_PARENT packet, is eligible to become a parent only when it does not want to sleep, i.e. SS(N(k)) is false, and it is currently alive, i.e. ALIVE(N(k)) is true. Once the node N(k) meets its eligibility criteria, it replies back a PARENT_REPLY packet with including information such as level number and remaining energy which are represented by LEVEL(N(k)) and RE(N(k)) respectively. Now, N(k) is ready to become parent of any node if it is chosen by a node N(j), which is looking for a parent. A node N(j) might receive multiple PARENT_REPLY packets from different $type_0$, $type_1$ or $type_2$ nodes. Out of all the PARENT_REPLY packets from its neighboring nodes, the node N(j) chooses the most suitable node N(k) to be its parent based on the following criteria.

- Node N(j) will give a higher priority to a node N(k) to be its parent if LEVEL(N(k)) is equal to LEVEL(N(j))-1 rather than LEVEL(N(k)) is equal LEVEL(N(j)).

Nevertheless, it will out-rightly reject all nodes N(k) if LEVEL(N(k)) > LEVEL(N(j)).

- Bsed on above criteria, If node N(j) has more than one possible nodes then it selects its parent whose remaining energy is high. The node N(j) chooses N(k1) over N(k2), if LEVEL(N(k1)) is equal to LEVEL(N(k2)) and RE(N(k1)) > RE(N(k2)).

After node N(j) selects its parent from the multiple PARENT_REPLY packets, it sends an ACKNOWLEDGEMENT message to node N(k). When node N(k) receives the ACKNOWLEDGEMENT packet, it acts depending on the type of node it is. If it is a $type_0$ node then it turns to $type_1$ node and sets its CHECK_REDUNDANCY(N(k)) to true. As this node turns active from sleep mode, some of its $type_2$ neighbors might have the possibility to become redundant. So, N(k) performs a redundancy check after a random amount of time among its $type_2$ neighbors, and all nodes found to be redundant are turned to $type_0$ nodes from $type_2$ and put to sleep mode. Once the redundancy check gets over, node N(k) becomes an internal node and turns to $type_1$ node and remains active for the $n + 1^{th}$ round. If N(k) is $type_2$ node then it turns to $type_1$ node as $type_1$ and $type_2$ are both coverage nodes Inter-conversion between these two types of nodes does not require a redundancy check as the coverage was already ensured up to n^{th} round. If node N(k) is a $type_1$ node then it remains a $type_1$ node and does not do any redundancy check as it had already been done before when it turned to $type_1$ node in previous rounds.

After this process gets over, the node N(i), who was waiting for a random amount of time, wants to check whether all of its children have received a new parent or not. It carries out a sequence of events that comprise of message passing to and from its neighbors to determine whether it has been able to get rid of all its children. Still, if N(i) has a child, then it cannot be $type_2$ for the $n + 1^{th}$ round as it has not been able to lose all its child nodes. So N(i) has to remain $type_1$ for the $n + 1^{th}$ round too. N(i) again tries to go to sleep for the $n + 2^{th}$ round after the completion of the $n + 1^{th}$ round. However, if every child N(j) of N(i) can receive a new parent, N(i) gets rid of all its children, and changes its state to $type_2$ node for the $n + 1^{th}$ round.

Once the parent selection process gets completed, all the $type_1$ nodes with parameter CHECK_REDUNDANCY(N(k)) as true check for redundant nodes among its $type_2$ neighbors and the newly found redundant nodes are converted to $type_0$ and put them to sleep mode. When the $n + 1^{th}$ round starts, CONS(N(i)) for all the $type_1$ nodes is increased by 1 to keep track of the no of consecutive rounds a node is active. After each nodes state gets decided, $type_1$ and $type_2$ nodes can resume its task of event detection and send the data packets to the sink through the intermediate nodes. For each packet being sent to the sink node by node N(i), PT(N(i)) is increased by 1.

After a certain number of tree reconstruction phase, when the RE(N(i)) reaches below the threshold energy, then node N(i) needs to die permanently and will not be a part of the tree anymore. However, if N(i) is allowed to die immediately, it might lead to a coverage hole. If node N(i) is a $type_0$ node, it can be allowed to die immediately as it does not provide coverage anyway. If node N(i) is a $type_2$ node and it was instrumental in providing coverage then letting it die instantly, leads to a void in coverage. So, node N(i) broadcasts a TURN_T_2 message among its neighbors, which instructs all its $type_0$ neighbors to turn to $type_2$ to fill the void in coverage. In addition to this, node N(i) broadcasts a CHK_RDNCY message, which is meant for its $type_1$ neighbors. This sets the CHECK_REDUNDANCY(N(k)) as true for the $type_1$ neighbor such as N(k). Then, node N(k) performs redundancy check among its newly turned $type_2$ neighbors, and it turns all the redundant $type_2$ nodes to $type_0$, which was converted because of TURN_T_2 message from node N(i). N(i) does not take part in the redundancy check process of N(k). Once this process gets over, full coverage is maintained without N(i) being a part of it. So ALIVE(N(i)) is set off, i.e., it can die now without any issue.

4. Simulation Results. The proposed protocol performances have been measured using one of the popular simulator exclusive for WSN, called Castalia [31]. Around 1000 number of sensor nodes have been deployed randomly in a deployed area of 250x250m^2 to analyze the proposed approach. As per the universal standard mentioned in the TelosB data sheet [26], the radio transmission power, sensing range, etc., have been considered for this simulation purpose.

To fulfill the primary objective of WSN, i.e., sense the whole deployed area along with maintaining network lifetime, the proposed level-wise tree construction approach includes the area coverage into consideration to fulfill the above two requirements. To validate the efficiency of the proposed algorithm with respect to network lifetime, in figure 4.1, we compare the number of nodes alive after each tree reconstruction round in the proposed

Algorithm 1: In The tree reconstruction phase between two round

Function *In TIMER_2:*
- **if** *N(i) is type$_1$ AND sleep constraint satisfied* **then**
  - starts the process to sleep;(set timer of start_process)
- **else**
  - do nothing until next round
- **if** *(ALIVE(N(i))==FALSE AND N(i) is type$_2$* **then**
  - Broadcast TURN_T_2 message;
  - Broadcast CHK_RDNCY message
- Trigger TIMER_3 after a random time;

Function *In start_process:*
- Send FIND_PARENT packet to notify N(j) (their child) that they need to find new parent;
- After a random time interval it triggers "determine_is*type$_1$*";

if *N(j) receives FIND_PARENT packet* **then**
- **if** *ALIVE(N(j)) AND [PFj,1==N(i) OR PFj,2==N(i)]* **then**
  - Broadcasts a WANT_PARENT packet;
  - Waits a random time while it chooses the best parent reply.

Function *In TIMER_3:*
- Trigger FIND_REDUNDANT function;
- Send packets to Sink node(if *type$_1$* and *type$_2$*);
- TIMER_2 after a certain time(next round);

if *N(k) receives WANT_PARENT packet* **then**
- **if** *(ALIVE(N(k)) == true and SS(N(k)) == false) and (LEVEL(N(k)) == LEVEL(N(j)) or LEVEL(N(k)) +1 == LEVEL(N(j))* **then**
  - broadcast PARENT_REPLY packet

if *N(j) receives PARENT_REPLY packet* **then**
- **if** *if N(j) wants new parent or PFj,1/PFj,2 is -1)* **then**
  - **if** *[LEVEL(N(k+1))==LEVEL(N(l))] OR [if(RE(N(k)) >RE(N(l))) AND (LEVEL(N(k))==LEVEL(N(l)))]* **then**
    - Select node N(k) as parent over N(l);
    - Send ACKNOWLEDGEMENT packet to its newly selected parent N(k)

if *N(k) receives ACKNOWLEDGEMENT packet* **then**
- **if** *N(k) is type$_1$ node* **then**
  - Remains *type$_1$*
- **if** *N(k) is type$_2$ node* **then**
  - Turns to *type$_1$*
- **if** *N(k) is type$_0$ node* **then**
  - CHECK_REDUNDANCY(N(K))=True;
  - turns to *type$_1$*

if *N(l) receives TURN_T_2 packet* **then**
- **if** *N(l) is type$_0$* **then**
  - turn to *type$_2$*

if *N(m) receives CHK_RDNCY packet* **then**
- **if** *N(l) is type$_1$* **then**
  - CHECK_REDUNDANCY(N(l))= True

Function *In determine_istype$_1$:*
- Determine number of child through a sequence of message **if** *no_of_child >0* **then**
  - remains *type$_1$*
- **else**
  - turns *type$_2$*

Function *In determine_istype$_1$:*
- **if** *CHECK_REDUNDANCY(N(i))= True* **then**
  - check redundancy among *type$_2$* neighbour nodes

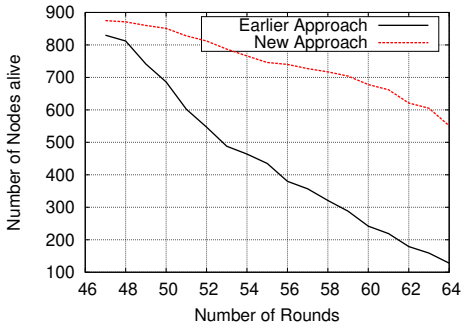


Fig. 4.1: Number of nodes alive after each tree reconstruction round in the normal level-wise tree construction approach [27] vs the proposed level-wise tree construction along with coverage.

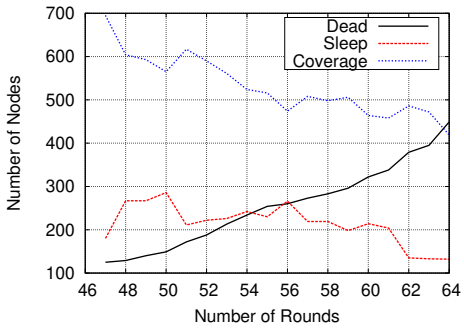


Fig. 4.2: Number of dead nodes, sleep nodes and coverage node after each each tree reconstruction rounds.

approach compared to the normal level-wise approach given [27]. From this figure, it is observed that after 46^{th} round, the nodes start dying out in both the earlier approach and the proposed approach. At 64^{th} round, in the previous approach, almost all nodes have died out, whereas in the case of the proposed method, after the same number of tree reconstruction rounds, the number of nodes dies out is nearly 50%. So, the network lifetime in the proposed approach is almost double as compared to the earlier approach. It indicates that the proposed method not only increases the network lifetime but also achieving area coverage.

Figure 4.2 shows that after the nodes start dying out, how the alive nodes become segregated into completely sleep node and coverage nodes in each tree reconstruction round. Here, the coverage nodes include both intermediate nodes, which are completely active, and the leaf nodes, which are in sleep mode. When an event occurs in their surroundings, those nodes wake up, complete the event detection process, and again go back to sleep mode. From this figure, it is observed that with an increase in the number of dead nodes, both the coverage node and sleep nodes decreases. Here, one interesting observation is that with the rise in the number of dead nodes, the area coverage is even managed with fewer nodes up to a certain tree reconstruction round. As per the figure, it is 64^{th} round. After onwards, even with the total number of alive nodes, the complete coverage could not be achieved.

Figure 4.3 depicts the number of nodes present in each level of the tree after the nodes start dying out as the tree reconstruction round progresses. From this figure, it is observed that the level 2 and 3 have the maximum number of nodes, and the rate of dying out of nodes in level 1 is more as compared to level 2, which is again greater than level 3. The nodes closer to the sink spend more time in data transmission and thus consume more energy than the nodes far away from the sink.

As per the proposed approach, when the $type_1$ nodes want to sleep, the children of those nodes try to select their new parents from the group of nodes, i.e., either a leaf node or a sleep node, or an intermediate node in the previous tree reconstruction round. Figure 4.4 shows that the number of sleep nodes, intermediate nodes,

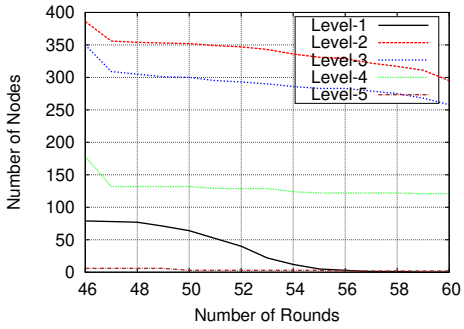


Fig. 4.3: Number of nodes present in each level of the tree in different tree reconstruction rounds.

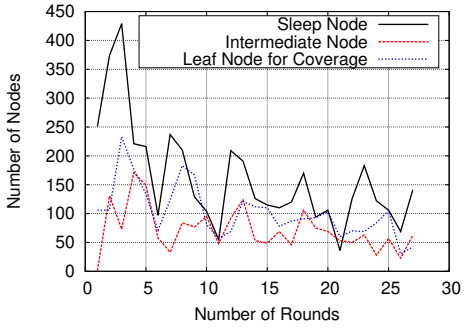


Fig. 4.4: Number of nodes select as parent in the next round of tree reconstruction from different group of nodes (sleep nodes, intermediate node, and leaf nodes).

and active nodes is selected as new parents by the nodes whose current parents are interested to sleep in the next tree reconstruction round. It is also observed that the nodes, which are looking for a parent, always have a higher tendency to choose $type_0$ nodes over $type_1$ & $type_2$ nodes.

Figure 4.5 depicts the number of $type_0$, $type_1$, and $type_2$ nodes in each level of tree at different round of tree reconstruction. In a network size of 1000 nodes, the number of levels comes up to 5. The majority of the nodes are dominated by $type_0$ and $type_2$ nodes. During the initial rounds of tree reconstruction, the number of $type_2$ nodes is more than $type_0$ nodes. As the round progresses, the tree adjusted to show a significant increase in $type_0$ nodes and a decrease in $type_2$ nodes.

In order to evaluate the longevity of the proposed protocol, we simulated both the earlier approach [27] as well as the proposed approach to identify the number of sleep nodes present in different rounds of the tree reconstruction as shown in figure 4.6. This figure ensures that the network lifetime achieved through the proposed approach is far better than the earlier level-wise tree construction approach. This enhancement is achieved as the coverage through the proposed approach is ensured with fewer nodes.

Figure 4.7 compares the number of sleep nodes, coverage leaf nodes, and intermediate nodes in different rounds of tree reconstruction. From this figure, it is observed that in the initial few rounds of tree reconstruction, $type_0$ nodes increase continuously, and coverage nodes decreased. Nevertheless, after the 46th round, more than 150 nodes died at once. So, there is a rise in the coverage nodes and drop in sleep nodes, as many sleep nodes turn to coverage nodes to compensate for the loss of so many coverage nodes.

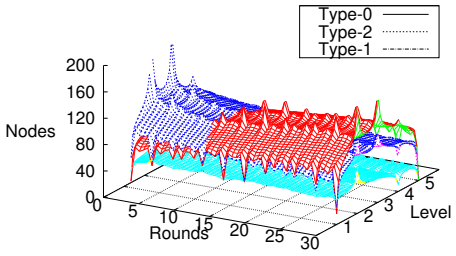


Fig. 4.5: Different types of nodes at different level of tree in different round tree reconstruction

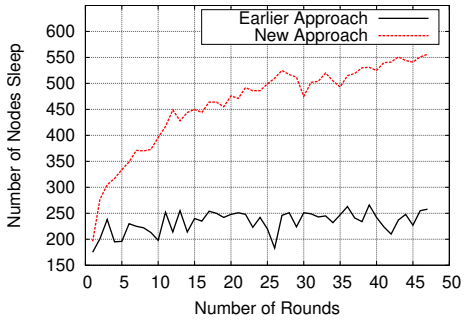


Fig. 4.6: Number of sleep nodes in different round of tree reconstruction in proposed approach vs. normal level-wise approach given in [27].

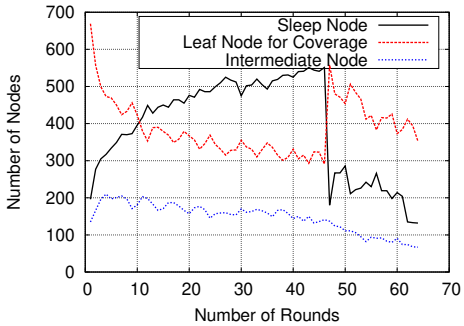


Fig. 4.7: Comparison of number of sleep nodes, coverage leaf nodes, and intermediate node in different round of tree reconstruction.

5. Conclusion. In this paper, we addressed the problem of scheduling sensor activities while having full coverage. Improving the network lifetime by increasing the number of sleep nodes along with maintaining full area coverage is the purpose of this research. We approached the least energy exhaustion provision in WSNs, proposed a level-wise sleep scheduling mechanism, and maintained coverage to achieve the above requirement. In this proposed approach, a routing tree is built that maintains coverage, and at the same time, longevity can be achieved by putting the redundant nodes in the sleep mode. The tree mends itself locally by following a level-wise approach for the tree reconstruction and finds the redundant nodes in the vicinity of the changing area only with the engagement of a minimum number of nodes in the process. Thus, full coverage is ensured only by engaging the nodes of the concerned area without involving all the nodes of the tree. The simulation results also ensured improved network lifetime by making more nodes in the sleep state.

REFERENCES

[1] J. Zheng and A. Jamalipour, Wireless Sensor Networks A Networking Perspective. A JOHN WILEY and SONS, Hoboken, New Jersey, 2009

[2] Z. Chuan, Z. Chunlin, S. Lei, H. Guangjie, A survey on coverage and connectivity issues in wireless sensor networks. J. Network Comput. Appl. 35, 619-632, 2012, Elsevier.

[3] Mulligan, R., Ammari, M.H., Coverage in wireless sensor networks: a survey. Network Protocols Algorithms 2 (2), 2010, 27-53.

[4] Amit, G., Sajal, D., Coverage and connectivity issues in wireless sensor networks: a survey. Sci. Direct, Pervasive Mob. Comput. 4, 303-334, 2008.

[5] Winston, J., Paramasivan, B., 2011. A survey on connectivity maintenance and preserving coverage for wireless sensor networks. Int. J. Res. Rev. Wireless Sensor Networks (IJRRWSN) 1 (2), 11-18.

[6] Fan Ye, F., Zhong, G., Cheng, J., Lu, S., Zhang, L., 2002. PEAS: a robust energy conserving protocol for long-lived sensor networks. In: 23rd International Conference on Distributed Computing Systems(DCS' 02), pp. 28-37.

[7] Gui, C., Mohapatra, P., 2004. Power conservation and quality of surveillance in target tracking sensor networks. In: Proceedings of 10th Annual International Conference on Mobile Computing and Networking (ACM MobiCom' 04), Pennsylvania, USA, pp. 129-143.

[8] Yen, Y., Hong, S., Hang, R., Chao, H., 2007. An energy efficient and coverage guaranteed wireless sensor network. In: Proceedings of IEEE Wireless Communications and Networking Conference, WCNC, pp. 2923-2930.

[9] More, A., Raisinghani, V., 2014. Random backoff sleep protocol for energy efficient coverage in wireless sensor networks. In: Kumar Kundu, M., Mohapatra, P.D., Konar, A., Chakraborty, A. (Eds.), Advanced Computing, Networking and Informatics, Wireless Networks and Security Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014), vol. 2. Springer, pp. 123-131.

[10] S. Slijepcevic and M. Potkonjak, "Power efficient organization of Wireless Sensor Networks", in Proc. of International Conference on Communications (ICC'01). IEEE, June 2001, pp. 472–476.

[11] C. F. Huang and Y. C. Tseng, "The coverage problem in a wireless sensor network," in Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '03), pp. 115–121, San Diego, Calif, USA, September 2003.

[12] Z. Zhou, S. Das, H. Gupta. "Connected K-Coverage Problem". Proceed- ings of the International Conference on Computer Communications and Networks (IC3N), 2004.

[13] H. Mousavi, A. Nayyeri, N. Yazdani and C. Lucas. "Energy conserving movement assisted deployment of Ad hoc sensor networks". IEEE Comm. Lett., 10(4): 269-271, 2006.

[14] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic. "Randomized Robot- assisted Relocation of Sensors for Coverage Repair in Wireless Sensor Networks". IEEE Comm. Lett, 2010.

[15] X. Li, G. Fletcher, A. Nayak, and I. Stojmenovic. "Randomized carrier based sensor relocation in wireless sensor and robot networks". Ad Hoc Networks. 11: 1951-1962, 2013.

[16] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sen- sor networks," in Proc. of IEEE International Conference on Communications (ICC), 2001, vol. 2, pp. 472–476.

[17] M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li and D.-Z. Du, "Wireless sensor networks with energy efficient organization," Journal of Inter- connection Networks, vol. 3, no. 3-4, pp. 213–229, 2002.

[18] D. Tian and N. D. Georganas, "A coverage preserving node scheduling scheme for large wireless sensor networks," in Proc. of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.

[19] Gao, Shan, Chinh T. Vu, and Yingshu Li. "Sensor scheduling for k-coverage in wireless sensor networks." International Conference on Mobile Ad-Hoc and Sensor Networks. Springer, Berlin, Heidelberg, 2006.

[20] Tian, Di, and Nicolas D. Georganas. "A coverage-preserving node scheduling scheme for large wireless sensor networks." Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. 2002.

[21] Cardei, Mihaela, et al. "Energy-efficient target coverage in wireless sensor networks." Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.. Vol. 3. IEEE, 2005.

[22] Yan, Ting, Tian He, and John A. Stankovic. "Differentiated surveillance for sensor networks." Proceedings of the 1st international conference on Embedded networked sensor systems. 2003.

[23] Alfieri, Arianna, et al. "Exploiting sensor spatial redundancy to improve network lifetime [wireless sensor networks]." IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.. Vol. 5. IEEE, 2004.
[24] Pottie, Gregory J., and William J. Kaiser. "Wireless integrated network sensors." Communications of the ACM 43.5 (2000): 51-58.
[25] Raghunathan, Vijay, et al. "Energy-aware wireless microsensor networks." IEEE Signal processing magazine 19.2 (2002): 40-50.
[26] Telosb data sheet. http://www.xbow.com/Products/ Product pdf files/Wireless pdf/TelosB Datasheet.pdf
[27] N. Tarasia, A. R. Swain, S. Roy and U. N. Kar, "A Level-Wise Periodic Tree ConstructionMechanism for Sleep Scheduling in WSN," Journal of Communications Software and Systems 16, no. 2 (2020): 113-121.
[28] Mhatre, Kavita Prashant, and Uday Pandit Khot. "Energy Efficient Opportunistic Routing with Sleep Scheduling in Wireless Sensor Networks." Wireless Personal Communications 112.2 (2020): 1243-1263.
[29] Fang, Wei, Mithun Mukherjee, Lei Shu, Zhangbing Zhou, and Gerhard P. Hancke. "Energy utilization concerned sleep scheduling in wireless powered communication networks," In 2017 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 558-563. IEEE, 2017.
[30] Guruprakash, B., C. Balasubramanian, and R. Sukumar. "An approach by adopting multi-objective clustering and data collection along with node sleep scheduling for energy efficient and delay aware WSN." Peer-to-Peer Networking and Applications 13.1 (2020): 304-319.
[31] *Castalia a simulator for wireless sensor networks*, http://castalia.npc.nicta.com.au/pdfs/Castalia User Manual.pdf.

Edited by: Dana Petcu
Received: Dec 21, 2020
Accepted: Jan 21, 2021

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2$_\varepsilon$ using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.